Microservice transition and its granularity problem: A systematic mapping study

Hassan, Sara; Bahsoon, Rami; Kazman, Rick

DOI:
10.1002/spe.v50.9

Document Version
Publisher's PDF, also known as Version of record

Citation for published version (Harvard):
Hassan, S, Bahsoon, R & Kazman, R 2020, 'Microservice transition and its granularity problem: A systematic mapping study', *Software: Practice and Experience*, vol. 50, no. 9, pp. 1651-1681. https://doi.org/10.1002/spe.v50.9

Received: 19 February 2020 | Revised: 16 May 2020 | Accepted: 17 May 2020

DOI: 10.1002/spe.2869

SURVEY PAPER

WILEY

Microservice transition and its granularity problem: A systematic mapping study

Sara Hassan[1] | **Rami Bahsoon**[1] | **Rick Kazman**[2,3]

[1]School of Computer Science, University of Birminhgam, West Midlands, UK

[2]Software Engineering Institute (SEI)/CMU, Pittsburgh, Pennsylvania, USA

[3]University of Hawaii, Honolulu, Hawaii, USA

Correspondence
Sara Hassan, Jeddah 2142, KSA.
Email: ssh195@cs.bham.ac.uk

Summary

Microservices have gained wide recognition and acceptance in software industries as an emerging architectural style for autonomic, scalable, and more reliable computing. The transition to microservices has been highly motivated by the need for better alignment of technical design decisions with improving value potentials of architectures. Despite microservices' popularity, research still lacks disciplined understanding of transition and consensus on the principles and activities underlying that transition. In this paper, we report on a systematic mapping study that consolidates various views, approaches and activities that commonly assist in the transition to microservices. The study aims to provide a better understanding of the transition; it also contributes a working definition of the transition and technical activities underlying it. We term the transition and technical activities leading to microservice architectures as microservitization. We then shed light on a fundamental problem of microservitization: microservice granularity and reasoning about its adaptation as first-class entities. This study reviews state-of-the-art and -practice related to reasoning about microservice granularity; it reviews modeling approaches, aspects considered, guidelines and processes used to reason about microservice granularity. This study identifies opportunities for future research and development related to reasoning about microservice granularity.

KEYWORDS

design decision support, granularity, microservices, software economics, systematic mapping study

1 | INTRODUCTION

Several industries have migrated their applications (or actively considering migrating) to microservices.[1-3] The transition to microservices has not been purely driven by technical objectives; the transition requires careful alignment of the technical design decisions with the business ones. The ultimate objective of this alignment is to enhance utilities of the application's software architecture and to improve its value potentials. For example, among the technical design decisions is isolating business functionalities into microservices that interact through standardized interfaces. Isolation motivated only by technical objectives can lead to aggressive decomposition of functionalities favouring service autonomy without considering its impact on value potentials. However, isolation motivated by technical and business objectives can be more

informed. It can enhance utilities such as autonomy, replaceability, and decentralized governance (among other utilities) to improve the microservice architecture's ability in coping with operation, maintenance, and evolution uncertainties. Ultimately, this can also relate to improved maintenance costs and cost-effective quality of service (QoS) provision to end users; these are examples of improved value potentials in the architecture.

Due to the recency of microservices, they have a multitude of definitions; each definition captures different properties of microservices. Definitions mostly agree that the fundamental properties of microservices include enabling facilitated improvement of component characteristics—autonomy, replaceability, independent deployability—and of architectural characteristics—improved reliability, scalability, resilience to failure, availability, and evolvability.[4-6] In essence, these definitions capture some drivers of the transition to microservices aimed at enhancing utility in the application's software architecture. The utility enhanced through the transition can render benefits that can cross-cut architectural design, testing, maintenance and service management.[7] For example, microservice autonomy allows the architects to easily locate, implement, and test necessary service amendments.[3] Microservice replaceability allows architects to confidently and independently add and/or manage new business functionalities over the system lifetime.[3]

The transition to microservices can help the application in better meeting its QoS requirements; this may consequently result into improved compliance with service level agreements for QoS, potential economics gains, and better alignment with the business objectives of microservice adopters.[3] Because of their "micro" character, microservices can be mobilized to the benefit of several service-oriented applications that can require "lighter weight" processing (eg, mobile services and Internet-of-Things [IoT]).[3]

Despite the industrial push toward microservices, there is no disciplined understanding of their transition nor consensus on the principles and activities underlying the transition.[8] A disciplined understanding of the transition is of paramount importance to inform and/or to justify its technical activities by aligning them with their added value and cost. Currently however, the state-of-the-practice in microservice adoption lacks appropriate methods and techniques that can justify value added of technical design decisions. For example, the software architect can be equipped with mechanisms and tools that can enhance replaceability by standardizing the communication paradigms across microservices.[9,10] Reasoning about the added value and possible cost becomes essential to justify this technical design decision regarding communication paradigms.

This paper is an attempt for a better understanding of the transition to microservices. It conducts a systematic mapping study to consolidate various views about microservices; it then uses the study results to contribute to a well-rounded working definition describing the transition and technical activities of the transition to microservice architectures. We term the transition and technical activities leading to microservice architectures as microservitization.

This working definition has explicitly considered a fundamental problem of microservitization: reasoning about the granularity of a microservice (ie, whether a microservice should be decomposed/merged further or not). A granularity level determines "the service size and the scope of functionality a service exposes (p. 426 of Reference 11)." Granularity adaptation entails merging or decomposing microservices thereby moving to a finer or more coarse grained granularity level.

Determining the granularity level too early in the software architecture's lifetime can lead to problems in reasoning about microservices.[12,13] This problem is of significance both in brownfield and greenfield development.[1] In both fields, a suitable granularity level is paramount to inform choosing concrete services from a plethora of commercial-off-the-shelf (COTS) microservices. Conducting a systematic mapping study to examine microservice literature is therefore essential to inform the paramount exercise of reasoning about granularity.

In this paper, we consider that "a systematic mapping study allows the evidence in a domain to be plotted at a high level of granularity[p. 5 of Reference 14]." "This allows for the identification of evidence clusters and evidence deserts to direct the focus of future systematic reviews and to identify areas for more primary studies to be conducted[p. 5 of Reference 14]." Directing the focus of future systematic reviews aligns with our aforementioned objectives. Ultimately, our attempt at defining the transition to microservices will pave the way for future development and research related to microservice transition. Furthermore, understanding the problem of reasoning about microservice granularity allows identifying areas for future primary studies. Since the examined literature regarding microservices spanned a broad variety of aspects, we found a systematic mapping study to be suitable given the amount of reviewed literature.[14]

In Section 2 we describe the steps we followed in the systematic mapping study. In Section 3 we report and briefly analyze our mapping study results. In Section 4 we use this analysis to: (i) present our working definition for the

transition to microservices (Section 4.1) and (ii) identify gaps in the state-of-the-art and -practice related to reasoning about microservice granularity (Section 4.2). Overall, the identified gaps motivate the need for:

- Microservice-specific modeling support potentially using an architecture definition language (ADL) that "treats microservice boundaries as adaptable first-class entities(p. 2 of Reference 15)" thereby facilitating runtime analysis of microservice granularity in a systematic architecture-oriented manner.[15]
- A dynamic architectural evaluation approach that captures two dimensions under uncertainty at runtime: added value to be introduced and cost to be incurred if the granularity of microservice architectures is adapted.
- Effective decision support that should systematically guide the software architects toward suitable granularity adaptation strategies at runtime or suggest revisiting their expectations of the architecture's runtime environment.

In Section 6 we compare and contrast-related literature reviews, studies and surveys in the field of microservices against our systematic mapping study. In Section 5 we reflect on threats to the validity of our study. In Section 7 we summarize contributions that are not directly linked to microservices but can be relevant to their emergence and development. In Section 8 we conclude by summarizing the results of our systematic mapping study.

2 | SYSTEMATIC MAPPING STUDY PROCESS

The process we follow in our mapping study is inspired by guidelines from Reference 14. In the subsections below, we describe our application of each stage of this process.

2.1 | Research questions

Overall, this paper conducts a systematic mapping study to address the following objectives:

- *Objective 1*: providing a better understanding of the transition to microservices—we consolidate various views (industrial, research/academic) of the principles, methods, and techniques that are commonly adopted to assist the transition to microservices. This consolidation allows us to reach a working definition for the transition to microservices; we term this transition *microservitization*.
- *Objective 2*: understanding a fundamental problem of the transition to microservices related to reasoning about their granularity—we review state-of-the-art and -practice related to reasoning about microservice granularity. This review allows us to understand the state-of-the-art and -practice in the modelling approaches, aspects considered, guidelines and processes used to reason about microservice granularity.

Given these objectives, we reify them into the following research questions. Along with each question we outline the rationale behind it.

Objective 1: Providing a better understanding of the transition to microservices

- What are the activities undertaken to adopt microservices? This question helps understand the principles, methods and techniques of the transition to microservices by digesting experiences of microservice adopters in industry and in academic research.

Objective 2: Understanding a fundamental problem of the transition to microservices related to reasoning about their granularity

- What are the modeling approaches used to define the granularity of a microservice? The aim of this question is to identify the support provided by microservice-specific models for reasoning about granularity and to investigate how systematic (ie, standardized and methodological) these models are.
- Which quality attributes are considered when reasoning about microservice granularity and how are they captured? The aim of this question is to elicit the possible trade-offs which software architects need to balance when reasoning about microservice granularity and/or how these trade-offs can be captured objectively.

- How is reasoning about microservice granularity described? The aim of this question is to explore the state-of-the-art regarding triggers and steps of microservice granularity adaptation and their suitability to the dynamic microservice environment.

2.2 | Search strategy

The terms used when searching for English publications were "microservice" and "micro-service"; Google Scholar, Association for Computing Machinery Digital Library, Institute of Electrical and Electronics Engineers (IEEE) Xplore, ScienceDirect, SpringerLink, and Wiley InterScience were used during this search. Our scope of publications includes journals, theses, books, conferences, workshops, blog articles, presentations, and videos. For academic publications, snowballing was applied[16] to further extract relevant literature. Nonacademic publications were included since most industrial experiences regarding microservices were published in these forms. We made our best effort, however, to only include articles that either transcribe the view of adopters or ones where the author is the opinion holder. We believe answering the research questions above comprehensively requires examining both academic and nonacademic experiences with microservices. This broad scope also aligns with a property of systematic mapping studies—aiming for broad coverage rather than narrow focus.[14] Our search was restricted to publications between January 2013 and April 2020, since the microservice trend had not emerged prior to that; nonacademic literature started to appear in 2013[17] while peer-reviewed publications started to appear in 2014.[18] Meta-data of the search results was maintained using a tool called "Publish or Perish.".[19]

2.3 | Selection of primary studies

Initially, the search results were examined for relevance according to inclusion and exclusion criteria below. Each research question elicited in Section 2.1 has corresponding inclusion/exclusion criteria (their structure is inspired by Reference 20). Along with each criterion is the rationale behind it. The rationale are presented in italics.

What activities are undertaken to adopt microservices?

Inclusion Criteria:

- Publications generically presenting the challenges of adopting microservices *since they can be used to infer the activities comprising the transition to microservices*.
- Case studies of adopting microservices *are used to complement and verify the activities in generic publications*.
- Publications comparing specific development tools in the microservice industry *because they can be used to infer activities comprising the transition*.

Exclusion Criteria:

- Publications without any reference to microservices. *Including such publications would confuse rather than clarify understanding the transition to microservices. This is against Objective 1 of our study*.
- Publications that refer to servitization in the business not the software context *since we are only concerned about the activities of shifting a software system from another architectural style to microservices*.

What modeling approaches are used to define the granularity of a microservice?

Inclusion Criteria:

- Publications defining formal notations/diagrams for modeling microservices. *Such publications can be used to assess how systematic the state-of-the-art is for modeling microservice granularity*.
- Proposals of modeling concepts for microservices. *Even when unverified, a proposed modeling concept can provide an insight for the building units of reasoning about microservice granularity*.
- Publications presenting industrial case studies for modeling microservices. *Such publications would verify and illustrate the expressiveness of proposed modeling concepts to microservice granularity*.

Exclusion Criteria:

- Papers which provide binding and reconfiguration solutions for service-oriented architectures (SOAs)/web-services/mobile services only *are excluded since they do not capture the properties specific to microservices, hence they are not suited for reasoning about microservice-specific decision problems (in this case microservice granularity)*.
- Papers which provide modeling approaches for SOAs/web-services/mobile services are *excluded since they do not capture properties specific to microservices, hence they are not suited for reasoning about microservice-specific decision problems (in this case microservice granularity)*.

Which quality attributes are considered when reasoning about microservice granularity and how are they captured?
Inclusion Criteria:

- Publications presenting metrics used when reasoning about microservice granularity. *Such publications would help assess how objectively the trade-offs affecting microservice granularity are captured in academia and/or industry*.
- Case studies involving the quality drivers considered when reasoning about granularity. *Such publications would realistically capture the significance of specific trade-offs when reasoning about granularity adaptation*.
- Publications focused on vendor-specific comparisons between platforms supporting reasoning about microservice granularity. *This can help derive the quality attributes and metrics considered when reasoning about granularity*.

Exclusion Criteria:

- Case studies that do not explicitly relate a challenge to its impact on microservice granularity. *Since case studies report concrete challenges and trade-offs impacting them, it is unreasonable to claim an impact of a reported trade-off on granularity if that is not reported explicitly in a case study*.

How is reasoning about microservice granularity described?
Inclusion Criteria:

- Publications including guidelines for reasoning about microservice granularity. *This helps to identify the state-of-the-art regarding triggers and/or steps for granularity adaptation*.
- Publications showing a sequence of when and how granularity is reasoned about in the application's lifecycle. *These publications help assess how much the state-of-the-practice considers dynamicity in microservice environments when reasoning about granularity adaptation*.

Exclusion Criteria:

- Publications that provide generic best practices for the granularity of applications with no reference to microservices (eg, related to web services, SOA, mobile services). *Such best practices are not targeted specifically at microservices, so it would not be reasonable to use them in the context of microservice granularity*.

2.4 | Keywords and classification

"The purpose of this stage is to classify papers with sufficient detail to answer the broad research questions and identify papers for later reviews without being a time consuming task[p44 of Reference 14]." Here we classify the included publications according to two classification frameworks. Initially, we classify them according to their research approaches. We elicit these approaches from the seminal guidelines in Reference 21.

The second classification framework entails categorizing the publications under categories derived from our research questions of concern. A publication belongs in a category if it contains any of the corresponding keywords identified in Table 1. The identification of the keywords is inspired by a microservice-specific systematic mapping study[13] and refined iteratively as more publications were examined. It is worth noting that for all the included publications, we manually

TABLE 1 Inferred classification framework used to classify the included publications in our study

Research question	Category	Keywords
What activities are undertaken to adopt microservices?	Architectural design	Architectural style, communication mechanism, boundaries, orchestration, service choreography, service registration, service discovery, design patterns, proxy, bulkhead, circuit breaker, router, routing
	Organization	Conway's law, decentralized governance, cross-functional teams, hierarchical teams
	Operation	Devops, NoOps, configuration settings, operation
	Deployment	Continuous integration, CICD, continuous deployment, deployment pipeline, automated deployment, virtualisation, hypervisors, containerisation, configuration provider
	Development	Heterogenous tools, agile development, extreme programming
	Monitoring	Regression unit testing, health monitoring, cluster monitoring, troubleshooting, debugging, failure
	Logging	central logging, decentralized logging, profiling, tracing
What modeling approaches are used to define the granularity of a microservice?	Structural	Domain-driven, classes, instances, resources, components, message format, data item, topology, service dependency, nodes, type definition
	Behavioral	Event flow, message stream, activity flow, communication flow, event triggers, execution timeline, use cases
Which quality attributes are considered when reasoning about microservice granularity and how are they captured?	Performance	QoS, efficiency, service contracts, SLA, performance, response time, throughput, performance bottleneck, invocation duration, transaction duration, customer value
	Reliability	Fault tolerance, disaster recovery, single point of failure, resilience, robustness, failure rate, error rate
	Scalability	Auto-scale, scalable, scaling, load balancing, load distribution, completed transactions per second
	Maintainability	Maintainable, changeable, maintenance cost, maintenance overhead, adaptability, changeability, effort cost, expandability, dynamicity
	Complexity	Communication overhead, complexity cost, development cost, tight coupling, low cohesion
How is reasoning about microservice granularity described?	Guidelines	Two-pizza team, lines of code, half-life, agile manifesto, one task, single responsibility, fine-grained functionality, separation of business concerns, high cohesion, loose coupling
	Processes	Iterative, strangler pattern

Abbreviations: CICD, continuous integration and continuous delivery; QoS, quality of service; SLA, service level agreement.

categorized them to ensure that synonyms or partial matches of these keywords are accurately handled. We justify the keywords (italicized below) under each category as follows:

What activities were undertaken to adopt microservices?

- Architectural design: publications in this category are concerned with all the technical activities which comprise adopting microservices. We use this category to verify whether or not microservice adopters call microservices an *architectural style* and/or *design pattern*. Choosing the generic *communication paradigm* of the architecture (eg, *orchestration, choreography*) and the more concrete message exchange pattern (eg, using a *router/proxy*) are also among the technical activities of microservice adoption. In addition, the *boundaries* of the system and individual components of the system is a technical design decision when moving to microservices. The fault tolerance mechanisms of the system also need to be identified (eg, *bulkhead* and *circuit breaker*patterns). Because microservice applications are highly distributed and scalable, two additional technical decisions are critical in microservitization: *service registration* and *service discovery*.

- Organization: in this category we are concerned with the organisational impact of adopting microservices. The state-of-the-practice in the microservice industry is to motivate *decentralised governance* (ie, holding the responsibility of building and running[22]) through microservitization.[4] The three most common means of achieving that is through following *Conway's law*[4]*, cross-functional teams, or hierarchical teams*.[23] Conway's law states "organisations which design systems … are constrained to produce designs which are copies of the communication structures of these organisations.[24]"
- Operation: publications in this category are concerned with identifying how the system will be governed post-deployment. This includes defining who is responsible for this governance to begin with. The state-of-the-practice is in the microservice industry is two alternative approaches: *DevOps* where the governance is shared across a development team and an operations team; and *NoOps*, where the governance is fully the responsibility of the development team. In addition, operational activities include defining the *configuration settings* and *configuration provider* which the governor of the microservice application needs to follow or in different runtime situations.
- Deployment: this category includes publications that define approaches for managing the *deployment pipeline* of a microservice application. The state-of-the-practice in the microservice industry is two alternative approaches:[25,26] continuous integration and continuous delivery (abbreviated as CICD), and *automated deployment*. "Continuous integration is a coding philosophy and set of practices that drive development teams to implement small changes and check in code to version control repositories frequently.[27]" "Continuous delivery picks up where continuous integration ends. Continuous delivery (CD) automates the delivery of applications to selected infrastructure environments.[27]" "Deployment automation allows applications to be deployed across the various environments used in the development process, as well as the final production environments.[28]" Common deployment automation tools in the microservice field include Kubernetes and Istio. Regardless of the deployment pipeline, the host on which this pipeline is enforced is another critical decision in this category. The three most common approaches for deploying microservices are *virtualization*, using *hypervisors,* and/or *containerization*. Cross-cutting these approaches are deployment activities that can be motivated by different quality attributes (eg, load-balancing and auto-scaling motivated by scalability). To answer the question of identifying which quality attributes are considered when reasoning about microservice granularity we find it more be-fitting to categorize these activities as keywords under their related quality attributes.
- Development: publications in this category describe how the transition to microservices impacts software development practices. The state-of-the-practice in microservice development is adopting *extreme programming and agile* practices using *heterogenous tools* (eg, Springboot, Fabric8).
- Monitoring: publications in this category are concerned with identifying the alternative rationales of runtime monitoring (eg, *health, cluster*) of microservice application and the alternative approaches which support monitoring (eg, *regression unit testing, troubleshooting, debugging,* and *failure* identification).
- Logging: in this category we are concerned with where the monitoring results are to be stored. Logging is alternatively called *profiling* or *tracing*. The two alternative approaches of logging in the microservice industry as we have examined are *central* or *decentralized logging*.

What modeling approaches are used to define the granularity of a microservice?

- Structural: publications in this category are concerned about contributions that capture the structure (alternatively called *topology*) of the microservice architecture. Depending on the nature of the contribution (eg, *domain-driven*[29]), the units of this structure differ (eg, *classes, instances, resources, components, data items, messages*, and/or *nodes*). Modeling these units includes capturing the *dependencies* across those units and their *types*.
- Behavioral: alternative to the approach above, publications in this category capture the sequence of actions (ie, *events, messages, activities, communication, execution steps,* and/or *use cases*) of a microservice application. The granularity of a microservice in this case would be defined in terms of which actions a single microservice is responsible for.

Which quality attributes are considered when reasoning about microservice granularity and how are they captured?

- Performance: wherever the main driving force of reasoning about granularity is performance (alternatively called *QoS, long-term value, efficiency or customer value*), the publication is put under this category. The metrics for capturing performance objectively include *response time, throughput, invocation duration*, identifying the *performance*

bottlenecks, and/or *transaction duration*. Thresholds on these metrics are captured in *service contracts, service level agreements*(abbreviated as SLAs). We subsume service contracts as "an agreement between the a consumer and provider service about the format of data that they transfer between each other. Normally, the format of the contract is defined by the consumer and shared with the corresponding provider. Afterwards, tests are being implemented in order to verify that the contract is being kept.[30]"

- Reliability: publications that imply or explicitly focus on enhancing reliability as the primary driving force when reasoning about granularity are included in this category. Reliability entails exhibiting *fault tolerance, disaster recovery, resilience*, eliminating *single points of failure,* and/or *robustness*. Reliability is captured objectively in terms of *failure/error rate*.
- Scalability: publications that reason about granularity in terms of how it enhances the scalability of the architecture are included in this category. Exhibiting scalability entails employing strategies such as *auto-scaling, load balancing,* and/or *load distribution*. Measuring scalability objectively can be done by looking at the *number of completed transactions per second* (or per unit of time more generally).
- Maintainability: exhibiting maintainability entails exhibiting *adaptability, changeability, expandability,* and/or *dynamicity*. These properties are objectively captured in terms of *maintenance cost, maintenance overhead*, and/or *effort cost*. Publications concerned with reasoning about granularity in terms of enhancing maintainability are included in this category.
- Complexity: minimizing complexity entails following two crucial design principles: *tight coupling*and/or *low cohesion*. This is measured in terms of *communication overhead, complexity cost*, and/or *development cost*. Publications where reasoning about granularity considers and/or measures its complexity are included in this category.

How is reasoning about microservice granularity described?

- Guidelines: publications under this category provide decision-making strategies for reasoning about granularity regardless of the steps of applying these strategies. The keywords under this category capture the state-of-the-practice guidelines.
- Processes: publications under this category are more elaborate in the sense that they enrich the granularity adaptation strategies with a sequence for applying them. The keywords under this category capture the alternative state-of-the-practice processes encountered for reasoning about microservice granularity.

2.5 | Data synthesis

Really Simple Syndication (RSS) feeds and manual search were used to obtain publications complying with the strategy defined in Section 2.2. The results were then manually examined for inclusion and categorised according to the criteria and frameworks described above (Sections 2.3 and 2.4, respectively).

3 | RESULT REPORTING AND ANALYSIS

In this section we present graphs summarizing distributions of the included publications along the categories described in Table 1. For each graph, we discuss how it helps serve the objectives outlined in Section 1.

3.1 | Publication distribution overview

A total of 877 publications met the inclusion criteria in Section 2.3. Table 2 lists representative examples of the included publications categorized according to Table 1. Figure 1 shows the overall distribution of the publications according to the publication type.

As justified in Section 2, we widened the scope of our study to include both academic and industrial publications. Broadly, we consider a publication type to be academic if it has gone through editing or peer-revision (the red bars in Figure 1); about 63% of the included publications. On the other hand, nonacademic publications account for about 36%

TABLE 2 Representative examples of publications included in the systematic mapping study

Research question	Category	Representative examples
What activities are undertaken to adopt microservices?	Architectural design	31-44
	Organization	45-54
	Operation	48,55-67
	Deployment	68-79
	Development	80-92
	Monitoring	93-99
	Logging	100-102
What modeling approaches are used to define the granularity of a microservice?	Structural	29,103-112
	Behavioral	8,113-117
Which quality attributes are considered when reasoning about microservice granularity and how are they captured?	Performance	118-124
	Reliability	125-130
	Scalability	131-135
	Maintainability	101,136-138
	Complexity	139-144
How is reasoning about microservice granularity described?	Guidelines	145-155
	Processes	110,156-164

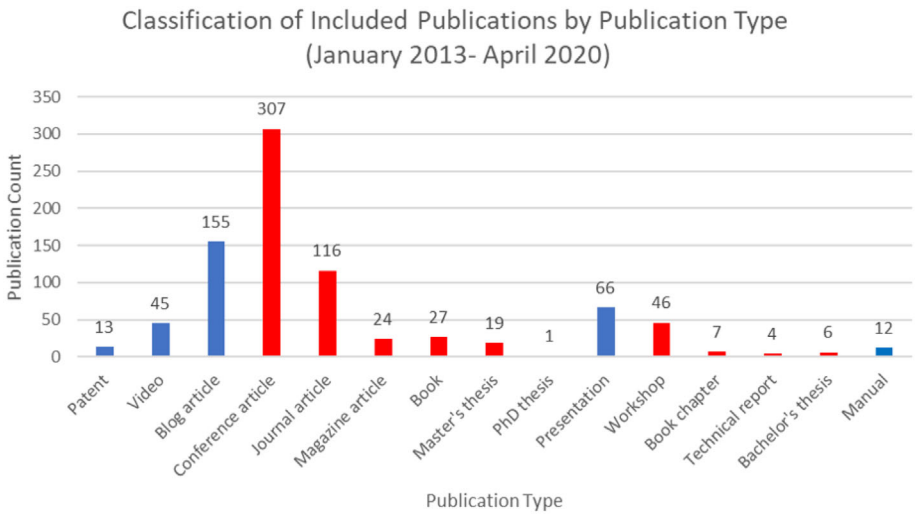

FIGURE 1 Publications between January 2013 and April 2020 as per the search strategy defined in Section 2.2 included as per criteria from Section 2.3 classified according to their publication type; the red bars are the publications we consider as academic and the blue bars are those we consider nonacademic [Colour figure can be viewed at wileyonlinelibrary.com]

of the total. Although the majority of publications are academic, nonacademic literature still comprises a significant percentage. Had we excluded these publications, attempting a definition for microservitization (Objective 1 in Section 1) would have been biased. The exclusion of nonacademic sources would lead to missing relevant keywords related to each category and the coverage of industrial opinions and experiences related to microservice adoption would be narrower.

We further classify peer-reviewed publications according to a highly cited paper classification framework[21] which targets IEEE papers. This framework classifies papers according to their research approach; a brief description of each approach is presented in Section 2. This framework has been applied before in the context of microservices,[13] so we consider it a neat fit for our study. To match the target context of the framework, we only apply it to peer-reviewed publications (Figure 2).

Solution proposals by far comprise the largest number of peer-reviewed publications. Solution proposals present novel, significant techniques without a full-blown validation. A proof-of-concept may be offered in solution proposals by

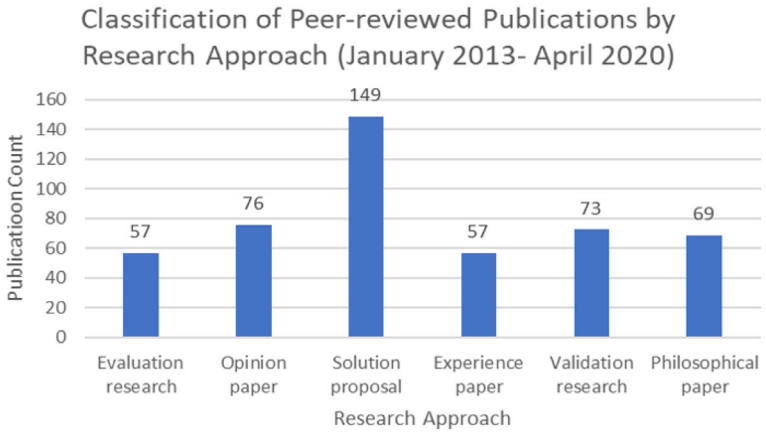

FIGURE 2 Academic (ie, peer-reviewed) included publications between January 2013 and April 2020 as per the search strategy defined in Section 2.2 classified according to their research approach; the classification criteria are derived from Reference 21 [Colour figure can be viewed at wileyonlinelibrary.com]

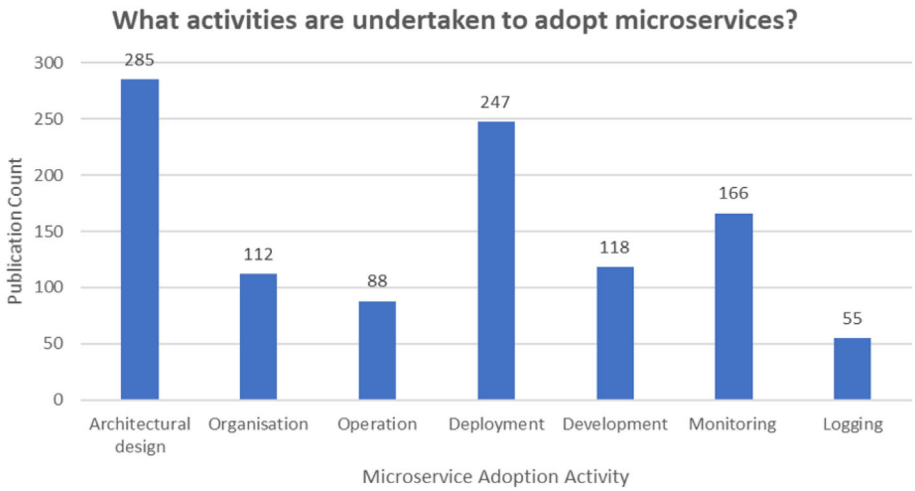

FIGURE 3 Included publications between January 2013 and April 2020 as per the search strategy defined in Section 2.2 that have keywords related to the first research question in Table 1 [Colour figure can be viewed at wileyonlinelibrary.com]

means of a small example, a sound argument, or by some other means.[21] Therefore, the microservices trend is a thriving field for novelty but it is still lacking maturity. Validation research publications — which thoroughly investigate solution proposals[21]—only amounted to 73 publications, which further proves the lack of maturity in the field. The large difference between the solutions proposals and validation research publications proves the need for disciplining the transition to microservices. The following subsections further discuss this need then focus on one of the fundamental problems of the transition—reasoning about microservice granularity.

3.2 | Objective 1: Providing a better understanding of the transition to microservices

Figure 3 classifies the publications which include keywords related to: *what are the activities undertaken to adopt microservices?* Architectural design and managing deployment are the most popular activities undertaken when adopting microservices. Therefore, we infer they are crucial activities in the transition to microservices. Nevertheless, there is a significant number of publications in the other categories of Figure 3. The variation in number of publications across categories of Figure 3 implies there is no consensus in describing the transition to microservices. This leaves room for us to contribute the microservitization term which attempts to provide a better understanding of the transition to microservices.

3.3 | Objective 2: Understanding the microservice granularity problem

One of the fundamental problems of the transition to microservices is finalising their level of granularity.[165] ADL classification frameworks[166] indicate that structural (or "topological[p26 of Reference 166]") as well behavioral aspects of an

architecture need to be modeled. Figure 4 helps to clearly identify the state-of-the-practice in modeling microservices; to answer *what are the modeling approaches used to define the granularity of a microservice?* The structural modeling approaches proposed for microservices are almost double the behavioral approaches. Structural approaches capture the topology and/or dependencies across building units of the microservice architecture. On the other hand, behavioral approaches capture the actions of these units in several runtime scenarios in which the modelled microservice operates. Therefore, a systematic, architecture-oriented modeling approach for microservice architectures which facilitates reasoning about granularity needs to capture the architecture's structural and behavioral aspects. The difference in numbers between structural- and behavioral-oriented publications in Figure 4indicates that there is a lack in modeling approaches that capture both aspects of a microservice architecture.

Figure 5 classifies publications according to the quality attributes they aim to optimise when reasoning about microservice granularity; to answer *which quality attributes are considered when reasoning about microservice granularity and how are they captured?* In other words, they can be the most common means to introduce utility through cost-effective microservitization. Scalability is the most common quality considered in the examined literature. This is reasonable given the dynamic, large-scale environment in which microservices operate.[8,167] Therefore, we infer that scalability can introduce added value to most microservice architectures. Relatively few publications have considered complexity/cost when reasoning about microservice granularity. Therefore, there is room for contributing to dynamic decision support which objectively considers both the potential value and cost of decisions related to microservice granularity (ie, adapting granularity by decomposing or merging microservices).

Figure 6 classifies proposed approaches for reasoning about microservice granularity according to how they are described; this answers *how is reasoning about microservice granularity described?* Most of the proposed approaches are ad-hoc guidelines that could be applied differently at various points of the microservice application's lifecycle. However, effective decision support for microservice granularity should comprise a clear sequence of distinct steps to be triggered under clear conditions. We have not found such effective support even in the 33 publications proposing a process to reason about microservice granularity. Therefore, we infer that there is a lack in effective decision support for reasoning about microservice granularity in terms of clear adaptation steps and triggers.

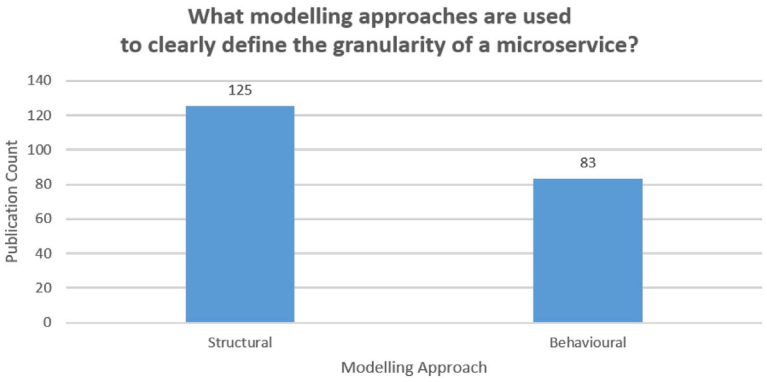

FIGURE 4 Included publications between January 2013 and April 2020 as per the search strategy defined in Section 2.2 which have keywords related to the second research question in Table 1 [Colour figure can be viewed at wileyonlinelibrary.com]

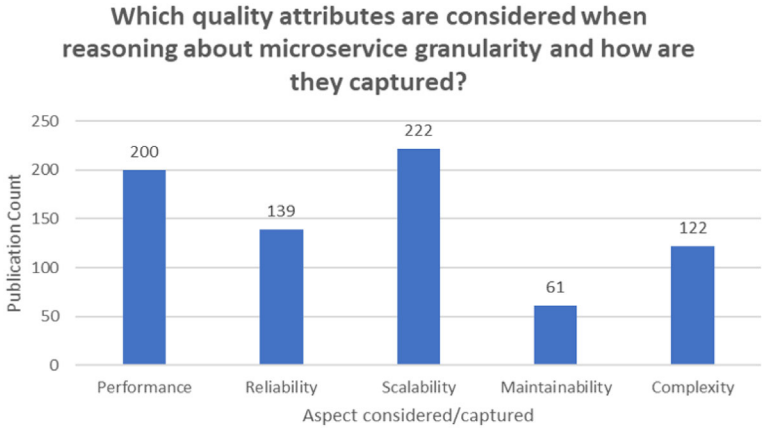

FIGURE 5 Included publications between January 2013 and April 2020 as per the search strategy defined in Section 2.2 that have keywords related the third research question in Table 1 [Colour figure can be viewed at wileyonlinelibrary.com]

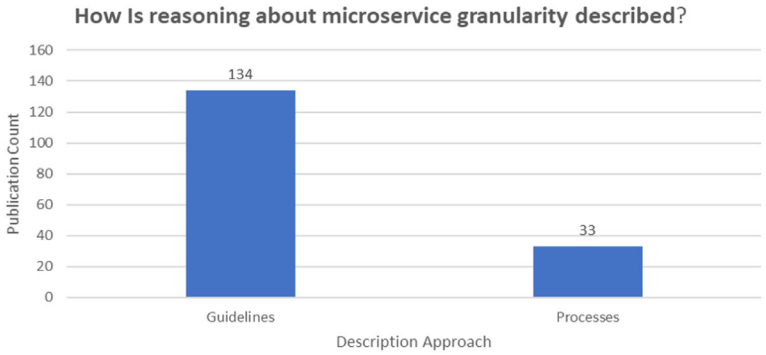

FIGURE 6 Included publications between January 2013 and April 2020 as per the search strategy defined in Section 2.2 that have keywords related the fourth research question in Table 1 [Colour figure can be viewed at wileyonlinelibrary.com]

4 | ADDRESSING SYSTEMATIC MAPPING STUDY OBJECTIVES

In Section 4.1 we contribute to an empirically grounded working definition for the transition to microservices; we call this transition microservitization. In Section 4.2, we identify the gaps in state-of-the-art and -practice related to reasoning about microservice granularity (inferred from Section 3) and discuss how they can be addressed.

4.1 | Objective 1: Providing a Better Understanding of the Transition to Microservices

We have not found in the surveyed publications an empirically grounded definition which characterises the transition to microservices, but rather several conflicting, informal attempts coming from industry and academia. Table 3 analyzes these attempts in terms of whether or not they explicitly include the activities derived from the microservice-specific systematic mapping study[13] and described in Section 2.4. It is worth noting that the attempts analyzed in this table are just a fraction of the publications summarized in Figure 3. In Table 3 we focus on the explicit attempts to define the transition

TABLE 3 Analyzing publications that attempt to define the transition to microservices included in the systematic mapping study; a T means the publication includes the activity in the corresponding column

	Microservice adoption Activity						
Publication	**Architectural design**	**Organization**	**Operation**	**Deployment**	**Development**	**Monitoring**	**Logging**
165	T	T		T			
168, 168				T			
169, 169	T	T	T				
170, 170	T					T	T
171, 171	T	T		T		T	T
172, 172				T			
173					T		
174	T	T				T	
175	T	T			T		
176	T		T	T	T	T	T
177	T						
18	T				T	T	
12		T		T			
178	T		T		T		
179				T	T		
153	T						

to microservices, while Figure 3 includes both explicit attempts to define the transition and case studies of microservice adoption which do not explicitly attempt to define the transition.

Observing Table 3, we introduce a definition for the transition to microservices (adapted from the activities included in Reference 13) which we call microservitization to cover all the relevant activities of the transition to microservices. This definition is adapted from activities included in Reference 13 and inspired by a trending concept in business and manufacturing domains[180] — servitization.

In the manufacturing and business domains, servitization is seen as a paradigm shift entailing "manufacturers growing their revenues and profits through services[181]" rather than tangible functional products. A service in this context is any feature that helps the business to (i) "really make money" and (ii) deliver new outcomes to customers. Examples of services in servitization include software applications, customer support, and self-service capabilities.[182]

Servitization "embraces business model innovation, organisational change, and new technology adoption. Services exist in various forms, and represent differing values to both the customer and provider[181]". To benefit from these values, servitization involves developing new relationships with customers, innovating customer value propositions, forming new value chain relationships, and adapting business models.[181,183] The key to successful servitization is "choosing the right technology, picking the appropriate moment to invest, and ensuring successful implementation[184]" which aligns with the business objectives.[182]

We liken the transition to microservices to servitization because of the following resemblances:

- Servitization is driven by delivering new outcomes to customers which can translate into revenues and profits. Similarly, the transition to microservices is driven by improving QoS provision to end users and translating that into an economic gain.
- Servitization entails embracing innovation in the manufacturing activities (eg, building business models and defining customer value propositions) are carried out. Similarly, the transition to microservices entails a dramatic change (motivated by value creation) to the way technical activities manifested in the software architecture are carried out.

Therefore, we define *microservitization as a form of servitization where "services/components are transformed into microservices — a more fine-grained and autonomic form of services* (p1 of Reference 15)" *— to introduce added value to the architecture*[3]. *"Microservitization is also an example of a paradigm shift*(p1 of Reference 15)" *since it involves dramatically changing how the following technical activities are carried out to align them with a microservice adopter's business objectives:*

- *Architectural design*: microservitization introduces the following critical architectural design activities:
  - Choosing light-weight communication mechanisms: microservitization can increase the distribution of functionalities across the architecture thereby introducing extra communication calls between microservices.[8] Therefore, rather than relying on enterprise service buses (which are the state-of-the-practice in SOA architectures), more light-weight mechanisms such as pipes and filters, event-based queues, and correlation identifiers are critical to avoid very high communication costs in microservice architectures. The large number of microservices can lead to a large volume of message exchange and hence high communication costs.
  - Reasoning about microservice granularity levels: a suitable granularity level is paramount to inform buying COTS concrete services or developing them in-house.[3] Choosing these services correctly is critical to introducing added value to the microservice architecture.
  - Adopting fault tolerance design patterns: although striving for fault tolerance is a best practice in any architecture, investing in fault tolerance design patterns is all the more critical to microservitization. The criticality is due to the scale of industries adopting microservices (eg, retail,[185] entertainment[61,186]) where microservices span different continents with a wide variety of end users. Microservice-specific fault tolerance design patterns include circuit breakers and bulkheads.
  - Incorporating microservice registration and discovery mechanisms: microservices are typically developed, deployed and replaced at a very quick rate.[8] Therefore, it is critical to incorporate robust registry and discovery mechanisms in microservice architectures to ensure an up to date record of the currently "alive" microservices.
- *Managing the organizational hierarchy*: microservitization has a direct impact on the organizational hierarchy.[187] In particular, the autonomy and independent deployability enhanced in the architecture through microservitization facilitate decentralised governance by breaking "silos" (based around strict separation of job roles) in the organization.
- *Operation*: microservitization aligns operation management with breaking organizational "silos." DevOps and NoOps are among the state-of-the-practice operation management approaches in microservitization; they involve "a set of

practices intended to reduce the time between committing a change to a system and the change being placed into normal production, while ensuring high quality[188]". Decentralised operation management can reduce the risk of bottlenecks that can materialise into economic losses to the microservice adopter. Reducing this risk is conditional upon development operation teams adhering to SLAs between them.

- *Deployment*: microservitization introduces a critical challenge of determining the hosts on which a deployment pipeline is implemented.[187] This is critical to balancing between the added value of microservitization and cost that can be introduced by a deployment pipeline implementation choice (eg, physical link installation, server rental and maintenance costs). Virtualization and containerization are among the common deployment pipeline implementation choices. They can enable swift auto-scaling the microservice architecture in response to changes in its runtime environment; this can materialize into economic gains for competitive microservice adopters.
- *Development*: unlike other software development paradigms, microservitization enables freedom in choosing development tools which can in turn introduce more added value to the architecture.[13] This freedom is a bi-product of the decentralised governance enabled by microservitization. It is worth noting that communication and knowledge sharing across teams using different development tools needs to be carefully managed to ensure this freedom actually introduces added value.
- *Monitoring*: microservitization requires much more robust, decentralised and customisable monitoring than that of classical SOAs due to the heterogeneity of tools used to develop microservices and scale at which they typically operate.[8] These requirements are critical to cater for the heterogeneity, scale, and dynamism of microservice architectures.
- *Logging*: maintaining logs of the monitoring data needs to be more customisable and distributable than logging classical SOAs due to the heterogeneity of tools used to develop microservices and scale at which they typically operate.[8] This requirements are aligned with the aforementioned monitoring challenges introduced by microservitization.

4.2 | Objective 2: Understanding the microservice granularity problem

Based on our definition above, microservitization introduces the challenge of reasoning about the suitable granularity level of a microservice.

To formalize the microservice granularity problem, the gap we identified is the lack of an architecture-oriented modeling approach that captures a microservice's granularity behaviour, thereby supporting runtime analysis of this behavior. The approach should treat microservice boundaries as the primitives for formulating the microservice granularity decision problem and actuators of microservice granularity adaptation decisions. These decisions include merging multiple microservices into a single boundary and decomposing a microservice into multiple ones encapsulated by multiple boundaries. In other words, this approach should treat microservice boundaries as adaptable first-class entities to ensure that both the structural and behavioral aspects of the microservice architecture are captured; we contribute to this approach in Reference 15. While conducting our study, we have seen architecture-oriented modeling approaches that treat the notion of boundaries statically (eg, Reference 189), or provide support for adaptability but without explicitly capturing the notion of adaptable boundaries (eg, Reference 190). Because the role of microservices is to encapsulate functionality, it is intuitive to use boundaries as adaptable first-class entities in the modelling approach. By contrast, if the decision problem was to determine the optimal physical infrastructure to host the microservice, for example, the adaptable first-class entity would be different (eg, microservice configuration variables).

To reason about microservice granularity objectively and dynamically, the gap we identified is the lack for an architectural evaluation approach that captures two aspects explicitly: added value to be introduced and cost to be incurred by pursuing granularity adaptation.

To effectively support reasoning about granularity adaptation, the gap we identified is the need for a decision support tool for reasoning about this problem at runtime. It is seen as a runtime problem since the suitable granularity level highly depends on the current scenario in which the microservice architecture is operating.[3] For example, if a certain functionality in a microservice-based application is continuously receiving a large volume of requests at runtime, it makes sense to decompose this functionality in a separate microservice to manage its load separately. On the other hand, if two microservices are continuously communicating across a network at runtime causing latency, then merging these microservices is sensible to help reduce such latency. Overall, uncertainties related to the expected environment and behavior[191] of the microservice architecture can not be fully captured at design time. Therefore, a runtime decision support tool is necessary to track and analyze this uncertainty. The tool should systematically guide the software architects

toward suitable granularity adaptation strategies at runtime or suggest revisiting their expectations of the microservice runtime environment. Each candidate granularity adaptation strategy must be systematically described as a sequence of merging/decomposition steps accompanied by triggers on them. Moreover, the tool's suggestions need to be justified objectively while leaving the final decision to the architects for adopting the suggested strategies; approaches such as Reference 192 can inspire the design of this tool.

5 | THREATS TO VALIDITY

In this subsection we acknowledge the threats to validity in the process we use in our systematic mapping study as well as the application of each stage. Threats to validity are "influences that may limit our ability to interpret or draw conclusions from the study's data (p351 of Reference 193)".

When defining our search strategy in Section 2.2, we considered blog articles, presentations, and videos as the means of reporting first-hand industrial experiences with microservice adoption. We acknowledge that an alternative means could be interviews with microservice adopters in the industry. However, published blog articles, presentations and videos are arguably more trusted since they present a more responsible and objective view than interviews. Though they can enrich the study with diverse opinions, interviews tend to suffer from bias, subjectivity, and irresponsible answers.[194] Moreover, our search strategy yields publications that explicitly mention microservices. Nevertheless, we acknowledge that there are publications prior without direct mention of microservices that could be relevant to their emergence and thereby to our research (eg, web service composition and agile development). We attempt to address this threat briefly in Section 7.

We acknowledge that the inclusion and exclusion criteria might have led to missing contributions that can inspire the microservices field. However, since our study was motivated by studying the state-of-the-art and -practice in the microservices trend we based the criteria on publications which already have a link to microservices. Nevertheless, we briefly outline the research areas that can inspire the microservice trend in Section 7. On a more technical level, we excluded publications that could not be translated to English which might have affected the study results.

We iteratively built Table 1 to include all the relevant keywords and made our best effort to justify them (Section 2). However, we acknowledge that some keywords might have been missed related to each research question.

When extracting videos and presentation for inclusion in the study results, we made our best effort to include videos whose content contained keywords from each category. The keywords are either mentioned by the speaker in the video or in the slides presented in the presentation. We acknowledge, however, that this might have biased the study results and that in the future transcription of the videos/presentations would be a more accurate means of determining their relevance.

When categorizing publications according to Table 1, we made our best effort to considers synonyms of the keywords in each category. However, since we categorized the publications by manual examination, we acknowledge that their distributions might have been skewed. In particular, we acknowledge that subjective interpretation of keywords might need to be complemented with more systematic approaches of categorizing the yielded publications. For example, the context in which a keyword or a fragment is mentioned needs to be considered before putting each publication under a certain category. A more systematic categorisation of the publications can ensure the reproducibility of our results.

We acknowledge that skewed distributions might lead to biassed inferences regarding gaps in the literature related to each objective of our study. For example, the variation in numbers of publications across categories in Figure 3 might be due to the inadequacies in keywords under each category. It might also indicate interest in architectural design across practitioners (ie, in nonacademic publications); this might not be as accurate for academic publications. Nevertheless, we argue that our mapping study results give a strong insight into the microservice trend and opens directions for more detailed research down each direction. For example, a systematic mapping study can be conducted which focusses specifically on microservice architectural design and/or development—the most dense categories in Figure 3.

6 | RELATED STUDIES

Motivated by disciplining the understanding of microservices, several studies have been conducted to examine the existing literature in this young yet trending field. They analyze existing literature with a variety of focuses. In this section, we compare and contrast the examined studies against the systematic study we conducted.

The closest to our study are References 13,37,195,196 since they both adopt a systematic mapping study process when examining the literature. However, the motivations in these studies are different from ours. In Reference 13, for example, the research questions motivating the study are related to challenges, modeling approaches and quality attributes considered when adopting microservices. These questions do overlap partially with both objectives of our study but they do not focus on reasoning about microservice granularity as we do in our study. In Reference 197, the composing microservices is studied from the perspective of semantic annotations. This overlaps slightly with the second objective of our study since composing microservices entails reasoning about granularity. However, our work takes a broader approach to studying the aspects of reasoning about granularity. In Reference 198, the activities comprising the transition to microservices are inferred through a literature review (aligned to Objective 1 of our study). However, it does not focus on microservice granularity so our study is significant to understanding this microservitization challenge. In Reference 199, a broad approach is taken to study microservice literature. We acknowledge that it goes beyond our work in studying the research approaches utilised in the microservice community. Furthermore, this study dives into the computing aspect of the transition to microservices (analogous to the development publications category in our study). Nevertheless, our study is goes beyond[199] with regards to defining the transition to microservices. Furthermore, our work is unique with regards to studying literature along different aspects of microservice granularity.

Several systematic literature reviews and surveys focus on defining the fundamental properties of microservices and the challenges of adopting them. They range in their rigour: some follow a rigorous search protocol[12,18,45,73,177,200-202] while others are less formal.[203] These studies overlap partially with Objective 1 of our study; they present activities related to microservices thereby contributing to a better understanding of the transition. However, they do not define on the transition to microservices nor can they be used to understand the microservice granularity problem. In Reference 175, the transition to microservices is partially described in terms of design patterns that can be applied to microservices. However, the scope of activities comprising the transition is not clearly defined. The approach of design patterns is also used in Reference 204. The patterns proposed in Reference 204, however, do not necessarily require microservice granularity adaptation. However, our work focuses on studying literature regarding microservice granularity. We envision that our work and References 175,204 can benefit from each other to elicit granularity adaptation patterns and to expand the concerns of microservice architectural design patterns.

Some studies mainly focus on modeling microservices.[205,206] Their focus partially overlaps with our following research question: *what are the modeling approaches used to define the granularity of a microservice?*

On the other hand,[207] aims to "construct knowledge of quality attributes in architecture through a systematic literature review (SLR), (an) exploratory case study and (an) explanatory survey (p1 of Reference 207)." In essence, this study can be used to partially address our question: *what are the quality attributes considered when reasoning about microservice granularity and how are they captured?* Nevertheless, the other research questions of our study were not answered in this study.

Overall, the examined studies can complement this paper to discipline the understanding of the transition to microservices. In this paper, our research questions are formulated with focus on a specific problem of this transition—reasoning about microservice granularity.

7 | RELEVANT RESEARCH FIELDS

Although we focus on publications that have direct links to microservices in our study, we acknowledge that there are publications prior to the time period considered in this mapping study that could be relevant to the emergence of microservices and thereby to our research. Such publications do not fit our search strategy because they do not make direct reference to microservices. Nevertheless, in this section we briefly summarize the examined literature in these areas indicating how they can be relevant to our systematic mapping study objectives.

7.1 | Research fields relevant to understanding the microservice transition

Even among microservice adopters, there are still debates over distinguishing SOAs from microservice architectures.[4,38,39,175,179,208-210] Therefore, we infer that contributions defining the properties and challenges of adopting SOA architectures can be relevant to understanding the transition to microservices.

Seminal work defines SOA as an architectural style which can guide business process definition[211-216] and support "rapid, low-cost composition of distributed applications (p1 of References 217-219)." The SOA style was introduced to address architectural complexity, redundant programming and inconsistent interfaces.[220,221] In SOAs a service is a self-describing unit that "consists of a contract, one or more interfaces and an implementation (p57 of Reference 222)." SOAs in turn comprise an application frontend, services, a service repository and enterprise service bus.

Despite the resemblance between microservices and SOAs, there is a subtle distinction we infer from our microservitization definition. The distinction comes from:[3] (i) the potential of microservices as autonomous fine-grained computational units with lightweight communication mechanisms rather than service buses and, (ii) the operational and organizational flexibility enhanced by microservitization. Further elaboration of these points is presented in Reference 223.

7.2 | Research fields relevant to understanding the microservice granularity problem

Modeling microservice granularity can be inspired by architectural modelling approaches—a wide research field, where contributions capture different notions of the architecture at varying levels of abstraction.[224] Domain-driven modeling is the most relevant to the modeling approach we describe in Section 4.2 because it strives for logical isolation of business functionalities.[29,225] However, domain-driven modeling is more concerned about the relationships between functional boundaries rather than their scope. In essence, domain-driven modeling can inform the structural rather than behavioral aspect of modeling microservices.

The Zachman framework[226] provides a comprehensive guide to the different dimensions and perspectives for architectural modeling. Two dimensions in this framework are aligned with the modeling approach we call in Section 4.2: *what* the model units are and *where* these units are located relative to each other. We call for a modeling approach that explicitly captures *what* each microservice is concerned about and helps define *where* the business functionalities encapsulated by the microservice are located relative to each other.

A more dynamic architectural modeling approach is feature modeling,[227] where an architecture is defined as a set of variability points, the candidates for each variability point and the rules constricting the dependencies across variability points. We appreciate this modeling technique is useful for formulating architectural decision-making problems. However, we would need to leverage such concept to focus on microservice boundaries being the variability point. While dependency rules in a feature model can give an insight about microservice granularity, they do not explicitly model it. ExecUtable RuntimE MegAmodels (EUREMA)[228] provides a yet more powerful dynamic modeling technique. Here an adaptation engine contains a runtime model which represents the evolution of the system as well as adaptation activities to be executed on that model. The link between the adaptation activities and the runtime model is expressed using runtime mega-models. Similar to feature modelling, EUREMA needs to capture the notion of microservice boundaries more explicitly.

Boundaries are modelled more explicitly in design structure matrices (DSMs).[229] Modularity metrics[230] can be used to assess the degree of interdependence across these boundaries. We have not seen a dynamic application of DSMs that captures the changes in these dependencies across a time unit (eg, release cycles).

Since we call for objective reasoning about microservice granularity adaptation, design metrics can inspire this requirement since they provide an objective way to capturing attributes of a design decision. Effort-based metrics[231] evaluate software development and maintenance efforts when a transition is made from centralised to distributed system architectures.[232] By analogy, an objective way is needed to evaluate development and maintenance costs when microservice granularity is adapted. Metrics related to cohesion, coupling and visibility of system components are presented and visualized in References 232,233, which can be used to assess the impact of granularity adaptation on the microservice architecture's modularity.

Since reasoning about microservice granularity is in essence a dynamic architectural design decision problem, several software engineering fields can be relevant to it. Architectural analysis methods, architectural design patterns, service composition and orchestration approaches, runtime architectural adaptation, architectural refactoring and feedback control loops are only some of the relevant fields. In the following subsections we categorise contributions in these fields according to their level of autonomy:

- *Manual* contributions with full reliance on the software architect and/or stakeholders (eg, architectural design patterns)

- *Partially autonomous* contributions where there is an autonomous agent but the software architect still makes the final decision regarding the optimal architecture (eg, interactive service orchestration and/or composition)
- *Fully autonomous* contributions where the decision-making process and executing the decision are fully handled by an autonomous agent (eg, feedback control loops, online architectural refactoring)

7.2.1 | Manual contributions

Out of the approaches in this subsection, the most cost- and value-aware approaches we examined are.[234-236] Refactoring the architecture according to patterns[234] or to introduce modularity[236] are regarded as value-bearing investments.[235] However, these approaches are only applied statically at design time. In this paper, we call for a similar view for reasoning about microservice granularity.

In Reference 237, a cost benefit analysis method (CBAM) is proposed as a generic architecture evaluation method which utilizes techniques in decision analysis, optimization, and statistics to evaluate architectural design decisions. However, CBAM does not dynamically track and update the added value of architectural decisions. These dynamic updates are critical to the nature of the microservice granularity problem.

In Reference 238, the net benefit of a software is calculated by deducting its total costs from the total benefit. These are manually elicited and monetised from software architects through a series of questions (eg, "what is the status of the environment without the system?"). To our knowledge, this approach, however, does not consider the uncertainty in the answers to these questions nor does it update them at runtime. In Reference 239 the predictive analysis of design captures the value-driven impact of architectural decisions as multidimensional normalized, weighted cost, and value vectors. Therefore, it can only provide static objective decision support for reasoning about microservice granularity.

The techniques in Reference 240 present different architectural evaluation methods and their motivations. Software Architecture Analysis for Evolution and Reusability (SAAMER), Scenario-Based Architecture Re-engineering (SBAR) and Architecture Level Prediction of Software Maintenance (ALPSM) in particular take an objective approach to architectural evaluation.

SBAR captures the runtime nature of decision-making by providing different quality attribute evaluation techniques depending on whether the quality attribute is concerned with the "development" of the system (ie, design time, such as reusability, which is handled by scenario-based evaluation) or the "operation" of the system (ie, runtime, such as performance, which is handled by simulation-based evaluation). SAAMER on the other hand partially addresses granularity problem by analyzing the level of interaction between different scenarios of the system as a means of assessing the level of functionality isolation in the system. Nevertheless, both methods have not been explicitly applied in a dynamic environment to our knowledge.

ALPSM and CBAM take a value-driven approach to the evaluation which makes them more systematic architectural evaluation approaches than SAAMER and SBAR. Furthermore, ALPSM uses probabilities to capture the likelihood of the impact of scenarios and CBAM captures the uncertainty the architectural analysis. ALPSM and CBAM therefore partially capture uncertainty, although they do not operate at runtime and thereby they suffer from the limitation of design-time analysis.

Classical design patterns presented in Reference 241 extensively study creational (concerned with object instantiation), structural (concerned with relationships between objects) and behavioral patterns (concerned with coordination between objects). These patterns are further categorized according to their static or runtime nature. In that context, reasoning about microservice granularity can benefit from runtime creational design patterns. However, the design patterns of that category in Reference 241 do not capture the scope—boundary—where a pattern can be enforced. Service workflow patterns have been presented in a seminal work[242] which implicitly discussed the issue of granularity is SOAs. However, we envision that the distinction between microservices and SOAs calls for explicitly addressing granularity adaptation decisions in the context of microservice constraints.

7.2.2 | Partially autonomous contributions

In References 243,244, pattern-based engines are proposed to synthesize a composition of atomic and composite services. However, we envision that reasoning about microservice granularity needs to be grounded on objective rather than

pattern-based evaluation. In References 245,246, a microservice-specific approach for addressing the microservice granularity problem is proposed which relies on microservice web application log mining to extract the usage pattern and then making adaptive decisions regarding microservice granularity to ensure an economically sustainable architecture. We acknowledge this work is very closely related to the decision support called for in this paper. Nevertheless, it does not explicitly analyze the value-driven implications of such adaptive decisions as we call for in Section 4.2.

7.2.3 | Fully autonomous contributions

The closest fully autonomous contribution to the effective decision support we call for in Section 4.2 is the ASTRO-CAptEvo orchestration framework.[247] It is a runtime framework that allows partial definition of business processes for service-based systems at design-time. Subsequently, the framework orchestrates "automatically composing the currently available services, provided by other actors and systems, according to the execution context and the goal of the process to be refined" using state transition systems. This framework takes a runtime approach to decision-making. However, the decision problem targeted by ASTRO-CAptEvo is service composition rather than microservice granularity. Moreover, ASTRO-CAptEvo does not consider objectively reason about composing the available services.

The Self-Serv framework presented in Reference 248 facilitates composite web service execution through peer-to-peer message exchange between coordination agents, which manage the service composition according to a static state chart. This framework can be utilised to address microservice granularity adaptation, but the knowledge that drives the service composition is static, meaning the runtime nature of the granularity problem is not captured in this framework.

Reputation-based dynamic service configuration techniques such as Reference 249 use a policy language to capture service consumers' and providers' profiles and then utilize these profiles to dynamically configure an optimal concrete service architecture. The work in Reference 250 leverages on the concept of reputation by capturing trust in the feedback given regarding the services. In particular, a model is proposed to aggregate feedback from several consumers of a service to reduce the effect of biased feedback. In both cases, a subjective user profile is used to drive the composition rather than an objective value- and cost-driven approach.

Case-based reasoning about concrete service composition is presented in Reference 251 where a solution space of composite services is formalized using recursive tuples of services. An agent then synthesises the optimal service decomposition given a request for services from the user which are then bound at runtime to concrete services fetched from a registry. A distinction is therefore made here between concrete service selection and the higher level composition of services; this can be utilised to address the granularity problem. However, this solution does not capture the dynamic nature of the microservice granularity problem.

Service composition techniques based on model checking[252-254] dynamically adapt probabilistic models of the system according to runtime changes in the scenarios surrounding the system or runtime changes in the requirements of the system. In Reference 254 Bayesian learning improves service composition synthesis process through runtime knowledge updates about the system's behavior over its lifetime. In Reference 253 an abstract service composition is mapped to a concrete service composition at runtime. The field of model-checking therefore is an attractive one for runtime decision-making. Such contributions however need to be leveraged for the specific problem we are concerned with (ie, the granularity of microservices).

Similar to the field of model checking is runtime architecture modeling. Several contributions in this field manifest runtime changes to the architecture.[228,255-263] Other contributions are catered for systems which exhibit a similar level of dynamism to microservices.[264-266] However, these contributions would need to be leveraged with objective reasoning that considers both the added value and cost of granularity adaptation.

Another approach of an autonomous solution is dynamic service formation rather than dynamic service composition. Frameworks such as References 267,268 provide means to dynamically produce web service specifications conforming to a service composition. These approaches can be utilized to complement the decision support we call for in this paper.

The runtime, uncertain context of microservice granularity adaptation calls for support similar to that provided by engineering self-adaptivity[3,191] into an architecture. The role of a self-adaptive solution is to refine and update at runtime the architects' design-time expectations about the architecture's behavior. There are several mechanisms which can be adopted in this solution.[269-273] Underlying most of them is the concept of feedback control loops[274,275] which can be used "to Monitor, Analyse, Plan and Execute adaptations[p16 of Reference 276]" in a system regarding trade-offs of concern (the phases of this loop are abbreviated as the MAPE loop). The knowledge learnt about the system needs to be maintained in a knowledge base (the phases of this loop are abbreviated as the MAPE-K[271]).

Control loops can be composed in a centralized, hierarchical, master-slave, or fully decentralized pattern.[273] Each pattern varies in which components of the architecture carry out which phase(s) of the control loop. The centralized pattern is more suitable for monolithic architectures. In a hierarchical pattern, the full MAPE loop is effected at individual services, with higher level services having a more general view of the architecture. The individual service MAPE loops pass information to the higher level loops at short time intervals. Although this pattern is well-suited to addressing the trade-offs of concern here, its only shortcoming is deciding what the higher level component with the global view of the architecture should comprise and the possibility of this service creating a bottleneck. A variant of the hierarchical pattern is the master-slave pattern where the individual services only monitor and execute while a higher level service comprises the analysis and planning phases. Although more lightweight than the hierarchical pattern, the master/slave pattern suffers from the same shortcomings as the hierarchical pattern. The decentralized pattern[277-280] on the other hand takes away the need for a service with a global view of the control loop. The MAPE loop is implemented in each service and information is passed across the services for decentralized management. The major challenge of this pattern is guaranteeing a consistent view of the system and its environment across all the control loops.[273] However, this pattern is the most aligned with the autonomy of microservice architectures and the scale at which microservices operate.

Runtime architectural adaptation approaches have been proposed before in the Rainbow framework[271] which is the most aligned with the concept of feedback loops. The Rainbow framework provides a reusable solution to induce self-adaptivity into a system in a cost-aware manner. However, it is debatable whether dynamically adapting the level of granularity of a microservice can be captured using the Rainbow framework. This is because the solution space here varies regarding the number of services used and the interaction patterns between them. To our knowledge, the Rainbow framework has not been applied to such a setting before. Another prominent approach is the architectural refactoring approach[281] where a set of anti-patterns is proposed which can be detected dynamically and used to trigger refactoring an architecture. This approach has as its main motive enhancing the modularity of the architecture rather than reasoning about modularity in a objective manner.

Realising microservice granularity adaptation involves changes to a deployed architecture. These activities are similar to those underlying the field of online architectural refactoring. Several contributions in this field pave the way to automated online architectural refactoring[282-288] and modeling transformations.[229,289] These contributions are driven by meeting a specific architectural design pattern, but they do not objectively reason about granularity adaptation.

In the field of AI planning, an ontology-based approach to architectural adaptation is proposed.[290] A shared ontology of generic "procedures" (or templates) is produced which the stakeholder can choose from at run-time. An agent then executes this procedure customizing it depending on the scenario in which the system will operate. It is appreciated that the use of ontologies promotes sharing knowledge across across architects. However, we envision that the decision support we are calling for in this paper can promote knowledge sharing and profiling to inform reasoning about microservice granularity.

8 | CONCLUSION

In this paper we report on a systematic mapping study to consolidate various views, principles, methods and techniques that are commonly adopted to assist the transition to microservices. We systematically describe the study's process and report its results. We contribute a working definition capturing the fundamentals of the transition; we term it as microservitization. Microservitization is a form of servitization[181,184] where services/components are transformed into microservices — a more fine-grained and autonomic form of services—to introduce added value to the architecture.[3] Microservitization is also an example of a paradigm shift since it involves a dramatic change to the way technical activities are carried out and aligned with a microservice adopter's business objectives. We then shed light on a fundamental problem of microservitization: microservice granularity and reasoning about its adaptation as first-class entities. This study has reviewed and identified gaps in the state-of-the-art and -practice that relate to the modeling approaches, aspects considered, guidelines, and processes used to reason about microservice granularity. The identified gaps pave the way to opportunities for future research and development related to reasoning about microservice granularity.

In particular, we identify there is room for future research regarding:

- A systematic architecture-oriented modeling support for microservice granularity that facilitates runtime analysis of microservice granularity in a systematic architecture-oriented manner.

- A dynamic architectural evaluation approach to reason about the cost and added value of granularity adaptation.
- Effective decision support to inform reasoning about microservice granularity at runtime.

Moreover, the findings of our study can be further disciplined in the future both horizontally and vertically. Horizontal disciplining entails including/excluding more microservitization activities (eg, securing communication links, cloud resource scheduling and data management) and more quality attributes related to reasoning about granularity (eg, security). Vertical disciplining entails using other evidence-based approaches which extract evidence from multiple sources (eg, interviewing microservice practitioners, industrial case studies).

ORCID

Sara Hassan https://orcid.org/0000-0001-7481-0434

REFERENCES

1. Dehghani Z, Zhamak dehghani real world microservices: lessons from the frontline. Youtube; 2015. https://youtu.be/hsoovFbpAoE.
2. Probst K, Becker J. Engineering trade-offs and the netflix API re-architecture; 2016. https://medium.com/netflix-techblog/engineering-trade-offs-and-the-netflix-api-re-architecture-64f122b277dd.
3. Hassan S, Bahsoon R. Microservices and their design trade-offs: a self-adaptive roadmap. Paper presented at: Proceedings of the 2016 IEEE International Conference on Services Computing (SCC); 2016:813-818. doi:https://doi.org/10.1109/SCC.2016.113.
4. Martin LJ. Microservices a definition of this new architectural term; 2014. http://martinfowler.com/articles/microservices.html.
5. Newman S. Microservices talk with Sam Newman; 2015. https://www.youtube.com/watch?v=GDVcUM5wbxU.
6. Newman S. Practical considerations for microservice architectures. Part 2. (Sam Newman, UK); 2014. https://www.youtube.com/watch?v=cU0J0w6sFoA.
7. Daya S, Van Duy N, Eati K, et al. *Microservices from Theory to Practice: Creating Applications in Ibm Bluemix Using the Microservices Approach*. Indiana, United States of America: IBM Redbooks; 2015 https://books.google.co.uk/books?id=eOZyCgAAQBAJ.
8. George F, Challenges in implementing microservices by Fred George; 2015.
9. OASIS AMQP is the internet protocol for business messaging; 2018. https://www.amqp.org/about/what.
10. Wagner T Microservices without the servers; 2015. https://aws.amazon.com/blogs/compute/microservices-without-the-servers/.
11. Kulkarni N, Dwivedi V. The role of service granularity in a successful SOA realization a case study. Paper presented at: Proceedings of the 2008 IEEE Congress on Services - Part I; 2008:423-430. doi:https://doi.org/10.1109/SERVICES-1.2008.86.
12. Kalske M, Mäkitalo N, Mikkonen T. Challenges when moving from monolith to microservice architecture. In: Garrigos I, Wimmer M, eds. *Current Trends in Web Engineering*. Cham: Springer International Publishing; 2018:32-47.
13. Alshuqayran N, Ali N, Evans R. A systematic mapping study in microservice architecture. Paper presented at: Proceedings of the 2016 IEEE 9th International Conference on Service-Oriented Computing and Applications; 2016.
14. Kitchenham B. *Guidelines for performing Systematic Literature Reviews in Software Engineering*. United Kingdom: Keele University and University of Durham; 2007.
15. Hassan S, Ali N, Bahsoon R. Microservice ambients: an architectural meta-modelling approach for microservice granularity. Paper presented at: Proceedings of the 2017 IEEE International Conference on Software Architecture (ICSA); 2017:1-10.
16. Petersen K, Vakkalanka S, Kuzniarz L. Guidelines for conducting systematic mapping studies in software engineering: an update. *Inf Softw Technol*. 2015;64:1-18. http://www.sciencedirect.com/science/article/pii/S0950584915000646, https://doi.org/10.1016/jinfsof.2015.03.007.
17. Trends G Microservices - explore; https://trends.google.com/trends/explore?date=2013-01-01&q=Microservices.
18. Francesco PD, Malavolta I, Lago P. Research on architecting microservices: trends, focus, and potential for industrial adoption. Paper presented at: Proceedings of the 2017 IEEE International Conference on Software Architecture (ICSA); 2017:21-30. doi:https://doi.org/10.1109/ICSA.2017.24.
19. Perish, Publish or Perish; 1999. https://harzing.com/resources/publish-or-perish.
20. Petersen K, Feldt R, Mujtaba S, Mattsson M. Systematic mapping studies in software engineering. Paper presented at: Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering Swindon, UK: BCS Learning & Development Ltd; 2008. p. 68–77. http://dl.acm.org/citation.cfm?id=2227115.2227123. doi:https://doi.org/10.1007/s00766-005-0021-6.
21. Wieringa R, Maiden N, Mead N, Rolland C. Requirements engineering paper classification and evaluation criteria: a proposal and a discussion. *Requir Eng*. 2005;11(1):102-107. https://doi.org/10.1007/s00766-005-0021-6.
22. Behara DGK, *Microservices Governance: A Detailed Guide*. Bonn, Germany: LeanIX; 2018. https://blog.leanix.net/en/microservices-governance.
23. Huston T, *What is Microservice Architecture?*. Massachusetts, United States of America: SmartBear; 2018. https://smartbear.com/learn/api-design/what-are-microservices/.
24. Conway ME. *How do Committees Invent*. California, United States of America: Datamation; 1968.
25. Wasson M, Tam B, *Designing Microservices: Continuous Integration*. Washington, United States of America: Microsoft; 2018. https://docs.microsoft.com/en-us/azure/architecture/microservices/ci-cd.

26. Ciuffoletti A. Automated deployment of a microservice-based monitoring infrastructure. *Proc Comput Sci*. 2015;68:163-172. http://www.sciencedirect.com/science/article/pii/S187705091503077X. https://doi.org/10.1016/j.procs.2015.09.232.
27. Sacolick I. What is CI/CD? Continuous integration and continuous delivery explained; 2018. https://www.infoworld.com/article/3271126/ci-cd/what-is-cicd-continuous-integration-and-continuous-delivery-explained.html.
28. ElectricCloud, Deployment Automation. http://electric-cloud.com/wiki/display/releasemanagement/Deployment+Automation#DeploymentAutomation-DeploymentAutomationOverview.
29. Evans EJ. *Domain-Driven Design: Tackling Complexity in the Heart of Software*. Boston, United States of America: Addison-Wesley Professional; 2004.
30. Novatec Introduction to microservices testing and consumer driven contract testing with PACT; 2017. https://blog.novatec-gmbh.de/introduction-microservices-testing-consumer-driven-contract-testing-pact/.
31. Namiot D. On micro-services architecture. *Int J Open Inf Technol*. 2014;09(2):24-27.
32. Krause L. Microservices: patterns and applications designing fine-grained services by applying patterns; 2015. https://books.google.co.uk/books?id=dd5-rgEACAAJ.
33. Haupt F, Leymann F, Scherer A, Vukojevic-Haupt K. A framework for the structural analysis of REST APIs. Paper presented at: Proceedings of the 2017 IEEE International Conference on Software Architecture (ICSA); 2017:55-58.
34. Haywood D, *In Defence of the Monolith, Part 1*. Toronto, Canada: InfoQ; 2017. https://www.infoq.com/articles/monolith-defense-part-1.
35. Miles R. Antifragile software building adaptable software with microservices. Leanpub; 2016.
36. Wolff E. *Microservices: Flexible Software Architecture*. London, United Kingdom: Pearson Education; 2016. https://books.google.co.uk/books?id=zucwDQAAQBAJ.
37. Pahl C, Jamshidi P. Microservices: a systematic mapping study. Paper presented at: Proceedings of the Proceedings of the 6th International Conference on Cloud Computing and Services Science - Volume 1 and 2 Portugal: SCITEPRESS - Science and Technology Publications, Lda; 2016:137-146. https://doi.org/10.5220/0005785501370146.
38. Nygard M. Release it! design and deploy production-ready software. *Release it! design and deploy production-ready software*. United States of America: Pragmatic Bookshelf; 2007.
39. Daya S, Van Duy N, Eati K, Ferreira CM, Glozic D, Gucer V. *Microservices from Theory to Practice: Creating Applications in IBM Bluemix Using the Microservices Approach*. Indiana, United States of America: IBM Redbooks; 2016 http://www.redbooks.ibm.com/abstracts/sg248275.html?Open.
40. Rodger R. The Tao of Microservices. Manning Publications Company; 2017. https://books.google.co.uk/books?id=uosOkAEACAAJ.
41. Fairbanks G. *Just enough software architecture: a risk-driven approach*. Colorado, United States of America: Marshall & Brainerd; 2010 https://books.google.co.uk/books?id=5UZ-AQAAQBAJ.
42. Rajasekar A, Wan M, Moore R, Schroeder W. Micro-services: a service-oriented paradigm for scalable, distributed data management. *Data Intensive Distributed Computing: Challenges and Solutions for Large-scale Information Management*. Pennsylvania, United States of America: IGI Global; 2012.
43. Cruz P, Astudillo H, Hilliard R, Collado M. Assessing migration of a 20-year-old system to a micro-service platform using ATAM. Paper presented at: Proceedings of the 2019 IEEE International Conference on Software Architecture Companion (ICSA-C); 2019:174-181.
44. Riahi T, Ashtiani M. A distributed and agent-oriented simulation framework based on the micro-service architecture; 2019.
45. Zimmermann O. Microservices tenets. *Comput Sci Res Dev*. 2017;32(3):301-310. https://doi.org/10.1007/s00450-016-0337-0.
46. Killalea T. The hidden dividends of microservices. *Commun ACM*. 2016;59(8):42-45. https://doi.org/10.1145/2948985.
47. Pautasso C, Zimmermann O, Amundsen M, Lewis J, Josuttis N. Microservices in practice, Part 1: reality check and service design. *IEEE Softw*. 2017;34(1):91-98. https://doi.org/10.1109/MS.2017.24.
48. Balalaie A, Heydarnoori A, Jamshidi P. Microservices architecture enables DevOps: migration to a cloud-native architecture. *IEEE Softw*. 2016;33(3):42-52. https://doi.org/10.1109/MS.2016.64.
49. Richardson C, Smith F, *Microservices: From Design to Deployment*. San Francisco, United States of America: Nginx; 2016.
50. O'Connor RV, Elger P, Clarke PM. Continuous software engineering — a microservices architecture perspective. *J Softw Evolut Process*. 2017;29(11):e1866. https://doi.org/10.1002/smr.1866.
51. Zhou X, Peng X, Xie T, et al. Benchmarking microservice systems for software engineering research. Paper presented at: Proceedings of the ICSE '18 Companion; 2018.
52. Dragoni N, Dustdar S, Larsen ST, Mazzara M. Microservices: migration of a mission critical system. *CoRR*. 2017;abs/1704.04173 http://arxiv.org/abs/1704.04173.
53. Wiggins A. *The Twelve-Factor App*; 2017.
54. Asik T, Selcuk YE. Policy enforcement upon software based on microservice architecture. Paper presented at: Proceedings of the 2017 IEEE 15th International Conference on Software Engineering Research, Management and Applications (SERA); 2017:283-287. doi:https://doi.org/10.1109/SERA.2017.7965739.
55. Betts T, *Q&A with Susan Fowler on Production-Ready Microservices*. Toronto, Canada: InfoQ; 2017. https://www.infoq.com/news/2017/01/production-ready-microservices.
56. Fisher T, *Digital Transformation is underpinned by the Enterprise IT Landscape*. Paris, France: Capgemini; 2014. https://www.capgemini.com/resources/digital-transformation-is-underpinned-by-the-enterprise-it-landscape/.
57. Fabrizio Montesi DST. Packaging microservices (work in progress). *IFIP International Federation for Information Processing*. New York, NY: Springer; 2017:131-137.

58. Bryant D. Shrinking microservices to functions: Adrian cockcroft discusses serverless at microXchg; 2017. https://www.infoq.com/news/2017/02/microxchg-microservice-functions.
59. Sanjeev-Shareema BC. *DevOps for Dummies*. 3rd ed. Indiana, United States of America: IBM Limited Edition. https://www.ibm.com/ibm/devops/us/en/resources/dummiesbooks/.
60. Curlett C, *Tame Microservices Complexity with APIs*. Massachusettes, United States of America: InfoWorld; 2016. https://www.infoworld.com/article/3111349/application-development/tame-microservices-complexity-with-apis.html.
61. Tonse S. Scalable microservices at netflix. *Challenges and Tools of the Trade*. Toronto, Canada: InfoQ; 2015. http://www.infoq.com/presentations/netflix-ipc.
62. Olliffe G, *Microservices: Building Services with the Guts on the Outside*; 2015. https://blogs.gartner.com/gary-olliffe/2015/01/30/microservices-guts-on-the-outside/.
63. Chen L. Microservices: architecting for continuous delivery and DevOps. Paper presented at: Proceedings of the IEEE International Conference on Software Architecture (ICSA 2018); 2018.
64. Florio L, Nitto ED. Gru: an approach to introduce decentralized autonomic behavior in microservices architectures. Paper presented at: Proceedings of the 2016 IEEE International Conference on Autonomic Computing (ICAC); 2016:357-362. doi:https://doi.org/10.1109/ICAC.2016.25.
65. Villamizar M, Garcés O, Ochoa L, et al. Infrastructure cost comparison of running web applications in the cloud using AWS lambda and monolithic and microservice architectures. Paper presented at: Proceedings of the 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid); vol 2016, 2016:179-182. doi:https://doi.org/10.1109/CCGrid.016.37.
66. Truong HL, Klein P. DevOps contract for assuring execution of IoT microservices in the edge. *Internet of Things*. 2020;9:100150 http://www.sciencedirect.com/science/article/pii/S2542660519301726. https://doi.org/10.1016/j.iot.2019.100150.
67. Taha MB, Talhi C, Ould-Slimanec H. A cluster of CP-ABE microservices for VANET. *Proc Comput Sci*. 2019;155:441-448. http://www.sciencedirect.com/science/article/pii/S1877050919309755. https://doi.org/10.1016/j.procs.2019.08.061.
68. Khazaei H, Barna C, Beigi-Mohammadi N, Litoiu M. Efficiency analysis of provisioning microservices. Paper presented at: Proceedings of the 2016 IEEE International Conference on Cloud Computing Technology and Science (CloudCom); 2016:261-268. doi:https://doi.org/10.1109/CloudCom.2016.0051.
69. Montesi F, Weber J. Circuit breakers, discovery, and API gateways in microservices. *CoRR*. 2016;abs/1609.05830 http://arxiv.org/abs/1609.05830.
70. Zheng T, Zhang Y, Zheng X, Fu M, Liu X. BigVM: a multi-layer-microservice-based platform for deploying SaaS. Paper presented at: Proceedings of the 2017 Fifth International Conference on Advanced Cloud and Big Data (CBD); 2017:45-50. doi:https://doi.org/10.1109/CBD.2017.16.
71. Brilhante J, Costa R, Maritan T. Asynchronous queue based approach for building reactive microservices. Paper presented at: Proceedings of the 23rd Brazillian Symposium on Multimedia and the Web; 2017:373-380; New York, NY, ACM. https://doi.org/10.1145/3126858.3126873.
72. Farcic V. The DevOps 2.0 toolkit automating the continuous deployment pipeline with containerized microservices. Leanpub; 2018.
73. Cerny T, Donahoo MJ, Trnka M. Contextual understanding of microservice architecture: current and future directions. *SIGAPP Appl Comput Rev*. 2018;17(4):29-45. https://doi.org/10.1145/3183628.3183631.
74. Wizenty P, Sorgalla J, Rademacher F, Sachweh S. MAGMA: build management-based generation of microservice infrastructures. Paper presented at: Proceedings of the 11th European Conference on Software Architecture: Companion; 2017:61-65; New York, NY, ACM. https://doi.org/10.1145/3129790.3129821.
75. Richter D, Konrad M, Utecht K, Polze A. Highly-available applications on unreliable infrastructure: microservice architectures in practice. Paper presented at: Proceedings of the IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C); vol 2017, 2017:130-137. doi:https://doi.org/10.1109/QRS-C.2017.28.
76. Toffetti G, Brunner S, Blöchlinger M, Dudouet F, Edmonds A. An architecture for self-managing microservices. Paper presented at: Proceedings of the 1st International Workshop on Automated Incident Management in Cloud; 2015:19-24; New York, NY, ACM. https://doi.org/10.1145/2747470.2747474
77. Zeiner H, Goller M, Expósito Jiménez VJ, Salmhofer F, Haas W. SeCoS: web of things platform based on a microservices architecture and support of time-awareness. *Elektrotechnik und Informationstechnik*. 2016;133(3):158-162. https://doi.org/10.1007/s00502-016-0404-z.
78. Rajavaram H, Rajula V, Thangaraju B. Automation of microservices application deployment made easy by rundeck and kubernetes. Paper presented at: Proceedings of the 2019 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT); 2019:1-3.
79. Höhr S, Tasci T, Verl A. Realization of data analytics projects in manufacturing using a microservice-based approach. Paper presented at: Proceedings of the 2019 IEEE International Conference on Mechatronics (ICM), vol. 1; 2019:321-326.
80. Porrmann T, Essmann R, Colombo AW. Development of an event-oriented, cloud-based SCADA system using a microservice architecture under the RAMI4.0 specification: lessons learned. Paper presented at: Proceedings of the IECON 2017 - 43rd Annual Conference of the IEEE Industrial Electronics Society; 2017:3441-3448. doi:https://doi.org/10.1109/IECON.2017.8216583.
81. Oksanych I, Shevchenko I, Shcherbak I, Shcherbak S. Development of specialized services for predicting the business activity indicators based on micro-service architecture. *Inf Technol*. 2017.2 84–95.
82. Sharma S. *Mastering Microservices with Java 9: Build Domain-Driven Microservice-Based Applications with Spring*. Birmingham, United Kingdom: Spring Cloud, and Angular Packt Publishing; 2017. https://books.google.co.uk/books?id=WsxPDwAAQBAJ.

83. Rajasekar A, Russell T, Coposky J, et al. *The integrated Rule-Oriented Data System (iRODS 4.0) Microservice Workbook*. 1st ed. California, United States of America: CreateSpace Independent Publishing Platform; 2015.
84. Mozaffari B. Microservice architecture building microservices with JBoss EAP 7. RedHat, 1.0 ed; 2016.
85. Terzić B, Dimitrieski V, Kordić S, Milosavljević G, Luković I. MicroBuilder: a model-driven tool for the specification of REST microservice architectures. *Enterprise Inf Syst*. 2017;12:1034–1057.
86. Esposte AMD, Kon F, Costa FM, Lago N. InterSCity: a scalable microservice-based open source platform for smart cities. Paper presented at: Proceedings of the 6th International Conference on Smart Cities and Green ICT; 2017.
87. Giallorenzo S, Lanese I, Mauro J, Gabbrielli M. Programming adaptive microservice applications: an AIOCJ tutorial. *Behavioural Types: From Theory to Tools*. Paris, France: HAL Archives; 2017:147.
88. Sorgalla J. AjiL: a graphical modeling language for the development of microservice architectures. *Extended Abstracts of the Microservices*; 2017.
89. Andrawos M, Helmich M. *Cloud Native Programming with Golang: Develop Microservice-Based High Performance Web Apps for the Cloud with Go*. Birmingham, United Kingdom: Packt Publishing; 2017. https://books.google.co.uk/books?id=NvNFDwAAQBAJ.
90. Ashikhmin N, Radchenko G, Tchernykh A. RAML-based mock service generator for microservice applications testing. In: Voevodin V, Sobolev S, eds. *Supercomputing*. Cham: Springer International Publishing; 2017:456-467.
91. Chen H, Chen P, Yu G. A framework of virtual war room and matrix sketch-based streaming anomaly detection for microservice systems. *IEEE Access*. 2020;8:43413-43426.
92. Lotz J, Vogelsang A, Benderius O, Berger C. Microservice architectures for advanced driver assistance systems: a case-study. Paper presented at: Proceedings of the2019 IEEE International Conference on Software Architecture Companion (ICSA-C); 2019:45-52.
93. Huttunen J. Micro service testing practices in public sector software projects; 2017. https://aaltodoc.aalto.fi/handle/123456789/26673.
94. Schaevitz S. *Deploying Changes to Production in the Age of the Microservice*. Dublin: USENIX Association; 2017 https://www.usenix.org/conference/srecon17europe/program/presentation/schaevitz.
95. Geerinck X, An architecture for resource analysis, prediction and visualization in microservice deployments; 2017. http://lib.ugent.be/fulltxt/RUG01/002/367/136/RUG01-002367136_2017_0001_AC.pdf.
96. Flygare R, Holmqvist A. Performance characteristics between monolithic and microservice-based systems; 2017.
97. Bryant D, *Observability and Avoiding Alert Overload from Microservices at the Financial Times*. Toronto, Canada: InfoQ; 2017. https://www.infoq.com/articles/observability-financial-times.
98. Janes A, Russo B. Automatic performance monitoring and regression testing during the transition from monolith to microservices. Paper presented at: Proceedings of the 2019 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW); 2019:163-168.
99. Schulz H, Angerstein T, Okanović D, van Hoorn A. Microservice-tailored generation of session-based workload models for representative load testing. Paper presented at: Proceedings of the 2019 IEEE 27th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS); 2019:323-335.
100. Salvadori I, Oliveira BCN, Huf A, Inacio EC, Siqueira F. An ontology alignment framework for data-driven microservices. Paper presented at: Proceedings of the 19th International Conference on Information Integration and Web-based Applications & Services; 2017:425-433. https://doi.org/10.1145/3151759.3151793.
101. Kothawade P, Bhowmick PS. Cloud security: penetration testing of application in micro-service architecture and vulnerability assessment; 2019:68.
102. Asenova M, Chrysoulas C. Personalized micro-service recommendation system for online news. *Proc Comput Sci*. 2019;160:610-615. http://www.sciencedirect.com/science/article/pii/S1877050919317399.
103. Ji Z, Liu Y. A dynamic deployment method of micro service oriented to SLA. *Int J Comput Sci Iss*. 2016;13:8–14.
104. Soenen T, Tavernier W, Colle D, Pickavet M. Optimising microservice-based reliable NFV management orchestration architectures. Paper presented at: Proceedings of the 2017 9th International Workshop on Resilient Networks Design and Modeling (RNDM); 2017:1-7. doi:https://doi.org/10.1109/RNDM.2017.8093034.
105. Uhle J, *On Dependability Modeling in a Deployed Microservice Architecture*. Potsdam, Germany: Universitat Potsdam; 2014.
106. Rademacher F, Sachweh S, Zündorf A. Towards a UML profile for domain-driven design of microservice architectures. In: Cerone A, Roveri M, eds. *Software Engineering and Formal Methods*. Cham: Springer International Publishing; 2018:230-245.
107. Nguyen P, Nahrstedt K. MONAD: self-adaptive micro-service infrastructure for heterogeneous scientific workflows. Paper presented at: Proceedings of the 2017 IEEE International Conference on Autonomic Computing (ICAC); 2017:187-196. doi:https://doi.org/10.1109/ICAC.2017.38.
108. Fowler M, *Bounded Context*. Illinois, United States of America: ThoughtWorks; 2014. https://martinfowler.com/bliki/BoundedContext.html.
109. Kaiser S, Podjarny G, Levine I, Burgess M, Stenberg J. *The Future of Microservices and Distributed Systems*. London, UK: QCon London Microservices Panel Discussion; 2018 https://www.infoq.com/news/2018/03/microservices-future.
110. Fowler M, *Event Interception*. Illinois, United States of America: ThoughtWorks; 2004. http://www.martinfowler.com/bliki/EventInterception.html.
111. Wang P, Dong L, Xu Y, Liu W, Jing N. Clustering-based emotion recognition micro-service cloud framework for mobile computing. *IEEE Access*. 2020;8:49695-49704.
112. Yang M, Huang M. An microservices-based openstack monitoring tool. Paper presented at: Proceedings of the 2019 IEEE 10th International Conference on Software Engineering and Service Science (ICSESS); 2019:706-709.

113. Richardson C. *Microservice Patterns*. New York, United States of America: Manning Publications Company; 2018. https://books.google.co.uk/books?id=UeK1swEACAAJ.
114. Fowler M, *Event Collaboration*. Illinois, United States of America: ThoughtWorks; 2006. https://martinfowler.com/eaaDev/EventCollaboration.html.
115. Baraiya V, Singh V, netflix conductor: a microservices orchestrator; 2016. https://medium.com/netflix-techblog/netflix-conductor-a-microservices-orchestrator-2e8d4771bf40.
116. Lv H, Zhang T, Zhao Z, Xu J, He T. The development of real-time large data processing platform based on reactive micro-service architecture. Paper presented at: Proceedings of the 2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC); vol. 1; 2020:2003–2006.
117. Gerking C, Schubert D. Component-based refinement and verification of information-flow security policies for cyber-physical microservice architectures. Paper presented at: Proceedings of the 2019 IEEE International Conference on Software Architecture (ICSA); 2019:61-70.
118. Niu Y, Liu F, Li Z. Load balancing across microservices. Paper presented at: Proceedings of the IEEE International Conference on Computer Communications; 2018.
119. Ismail U, Rolnick D, Fabijan D, Wallgren A. Challenges of micro-service deployments; 2016. http://techtraits.com/microservice.html.
120. Kukade PP, Kale PG. Auto-scaling of micro-services using containerization. *Int J Sci Res (IJSR)*. 2015;4(9).1960–1963.
121. Bryant D, *The Economics of Microservices: Phil Calçado Recommends Avoiding Microliths*. Toronto, Canada: InfoQ; 2017:
122. Shoumik FS, Talukder MIMM, Jami AI, Protik NW, Hoque MM. Scalable micro-service based approach to FHIR server with golang and No-SQL. Paper presented at: Proceedings of the 2017 20th International Conference of Computer and Information Technology (ICCIT); 2017:1-6. doi:https://doi.org/10.1109/ICCITECHN.2017.8281846.
123. Sami H, Mourad A, El-Hajj W. Vehicular-OBUs-as-on-demand-fogs: resource and context aware deployment of containerized micro-services. *IEEE/ACM Trans Netw*. 2020;28(2):778-790.
124. Akbulut A, Perros HG. Performance analysis of microservice design patterns. *IEEE Internet Comput*. 2019;23(6):19-27.
125. Naily MA, Setyautami MRA, Muschevici R, Azurat A. A framework for modelling variable microservices as software product lines. In: Cerone A, Roveri M, eds. *Software Engineering and Formal Methods*. Cham: Springer International Publishing; 2018:246-261.
126. Derakhshanmanesh M, Grieger M. Model-integrating microservices: a vision paper. Paper presented at: Proceedings of the Software Engineering Workshops.
127. Rajagopalan S, Jamjoom H. App–bisect: autonomous healing for microservice-based apps. Paper presented at: Proceedings of the 7th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 15); 2015; Santa Clara, CA, USENIX Association. https://www.usenix.org/conference/hotcloud15/workshop-program/presentation/rajagopalan.
128. Brogi A, Canciani A, Neri D, Rinaldi L, Soldani J. Towards a reference dataset of microservice-based applications. In: Cerone A, Roveri M, eds. *Software Engineering and Formal Methods*. Cham: Springer International Publishing; 2018:219-229.
129. Song Z, Tilevich E. Equivalence-enhanced microservice workflow orchestration to efficiently increase reliability. Paper presented at: Proceedings of the 2019 IEEE International Conference on Web Services (ICWS); 2019:426-433.
130. Ma M, Lin W, Pan D, Wang P. MS-Rank: multi-metric and self-adaptive root cause diagnosis for microservice applications. Paper presented at: Proceedings of the2019 IEEE International Conference on Web Services (ICWS); 2019:60-67.
131. Kecskemeti G, Marosi AC, Kertesz A. The ENTICE approach to decompose monolithic services into microservices. Paper presented at: Proceedings of the2016 International Conference on High Performance Computing Simulation (HPCS); 2016:591-596. doi:https://doi.org/10.1109/HPCSim.2016.7568389.
132. LOFTIS H, *Why Microservices Matter*. San Francisco, United States of America: Heroku; 2015. https://blog.heroku.com/why_microservices_matter.
133. Hasselbring W. Microservices for scalability: keynote talk abstract. Paper presented at: Proceedings of the 7th ACM/SPEC on International Conference on Performance Engineering; 2016:133-134; New York, NY, ACM. https://doi.org/10.1145/2851553.2858659.
134. Klinaku F, Frank M, Becker S. CAUS: an elasticity controller for a containerized microservice. Paper presented at: Proceedings of the Companion of the 2018 ACM/SPEC International Conference on Performance Engineering; 2018:93-98; New York, NY, ACM.
135. Alipour H, Liu Y. Online machine learning for cloud resource provisioning of microservice backend systems. Paper presented at: Proceedings of the2017 IEEE International Conference on Big Data (Big Data); 2017:2433-2441. doi:https://doi.org/10.1109/BigData.2017.8258201.
136. Oksa M, Web API development and integration for microservice functionality in web applications; 2016. http://urn.fi/URN:NBN:fi:jyu-201612215220.
137. Acevedo CAJ, y Jorge JPG, Patino IR. Methodology to transform a monolithic software into a microservice architecture. Paper presented at: Proceedings of the 2017 6th International Conference on Software Process Improvement (CIMPS); 2017:1-6. doi:https://doi.org/10.1109/CIMPS.2017.8169955.
138. Galbraith K. 3 methods for microservice communication; 2019. https://blog.logrocket.com/methods-for-microservice-communication/.
139. Müssig D, Stricker R, Lässig J, Heider J. Highly scalable microservice-based enterprise architecture for smart ecosystems in hybrid cloud environments. Paper presented at: Proceedings of the 19th International Conference on Enterprise Information Systems - Volume 3: ICEIS, INSTICC; 2017:454-459; SciTePress. doi:https://doi.org/10.5220/0006373304540459.
140. Ulander D, Software architectural metrics for the scania internet of things platform: from a microservice perspectiv; 2017. http://www.diva-portal.org/smash/record.jsf?pid=diva2:1115342.

141. Cockroft A, *Gluecon Monitoring Microservices and Containers: A Challenge*. Washington, United States of America: Slideshare; 2015. http://www.slideshare.net/adriancockcroft/gluecon-monitoring-microservices-and-containers-a-challenge.
142. Kleindienst P, *Implementation and Evaluation of a Hybrid Microservice Infrastructure*. Stuttgart, Germany: Stuttgart Media University; 2017.
143. Srirama SN, Adhikari M, Paul S. Application deployment using containers with auto-scaling for microservices in cloud environment. *Journal of Network and Computer Applications*. 2020;160:102629. http://www.sciencedirect.com/science/article/pii/S108480452030103X. https://doi.org/10.1016/j.jnca.2020.102629.
144. Kallergis D, Garofalaki Z, Katsikogiannis G, Douligeris C. CAPODAZ: a containerised authorisation and policy-driven architecture using microservices. *Ad Hoc Networks*. 2020;104:102153. https://doi.org/10.1016/j.adhoc.2020.102153.
145. Bryant D, *Microservices: The Organisational and People Impact*. Copenhagen, Denmark: Goto; 2017. https://gotocph.com/2017/sessions/295.
146. Vlaovic S, Pilani R, Parulekar S, Handa S, *Netflix Billing Migration to AWS*. California, United States of America: Netflix; 2016. https://medium.com/netflix-techblog/netflix-billing-migration-to-aws-451fba085a4.
147. Reinhold E, *Lessons Learned on Uber's Journey into Microservices*. Toronto, Canada: InfoQ; 2016. https://www.infoq.com/presentations/uber-darwin.
148. Simons D, *Decoupled APIs through Microservices*. Toronto, Canada: InfoQ; 2016. https://www.infoq.com/presentations/api-microservices-tools.
149. Richardson C, *A Pattern Language for Microservices*; 2014. http://microservices.io/patterns/index.html.
150. Sampaio AR, Kadiyala H, Hu B, et al. Supporting microservice evolution. Paper presented at: Proceedings of the 2017 IEEE International Conference on Software Maintenance and Evolution (ICSME); 2017:539-543.
151. Linthicum DS. Practical use of microservices in moving workloads to the cloud. *IEEE Cloud Comput*. 2016 Sept;3(5):6-9. https://doi.org/10.1109/MCC.2016.114.
152. Berger C, Nguyen B, Benderius O. Containerized development and microservices for self-driving vehicles: experiences best practices. Paper presented at: Proceedings of the 2017 IEEE International Conference on Software Architecture Workshops (ICSAW); 2017:7-12. doi:https://doi.org/10.1109/ICSAW.2017.56.
153. Stenberg J, *Strategies for Decomposing a System into Microservices*. Toronto, Canada: InfoQ; 2018. https://www.infoq.com/news/2018/06/decomposing-system-microservices.
154. Ma SP, Fan CY, Chuang Y, Liu IH, Lan CW. Graph-based and scenario-driven microservice analysis, retrieval, and testing. *Future Generat Comput Syst*. 2019;100:724-735. http://www.sciencedirect.com/science/article/pii/S0167739X19302614, https://doi.org/10.1016/j.future.2019.05.048.
155. Kousiouris G, Tsarsitalidis S, Psomakelis E, et al. A microservice-based framework for integrating IoT management platforms, semantic and AI services for supply chain management. *ICT Express*. 2019;5(2):141-145. http://www.sciencedirect.com/science/article/pii/S2405959519301158.
156. Fowler M, *Strangler Application*. Illinois, United States of America: ThoughtWorks; 2004. https://www.martinfowler.com/bliki/StranglerApplication.html.
157. Richardson C, Wolff E, Miles R, Kaiser S, Newman S, Tilkov S. microXchg; 2016. https://www.youtube.com/watch?v=wHuI7C3-Eis&list=PLx2By31njbhrs8caX08BEusyD_fBDu-XG&feature=player_detailpage.
158. Penchikala S, *Susanne Kaiser on Microservices Journey from a Startup Perspective*. Toronto, Canada: InfoQ; 2017. https://www.infoq.com/news/2017/07/kaiser-microservices-journey.
159. Posta C, *The Hardest Part of Microservices: Calling Your Services*. Arizona, United States of America: Christian Posta; 2017. http://blog.christianposta.com/microservices/the-hardest-part-of-microservices-calling-your-services/.
160. Posta C, *Low-risk Monolith to Microservice Evolution Part II*. Arizona, United States of America: Christian Posta; 2017. http://blog.christianposta.com/microservices/low-risk-monolith-to-microservice-evolution-part-ii/.
161. Shoup R, *GOTO 2016 - Pragmatic Microservices - Randy Shoup*. Copenhagen, Denmark: GOTO; 2016. https://youtu.be/9vS7TbgirgY.
162. Iffland D, *Q&A with Intuit's Alex Balazs*. Toronto, Canada: InfoQ; 2016. https://www.infoq.com/articles/intuit-alex-balazs-node-services.
163. Schäffer E, Mayr A, Fuchs J, Sjarov M, Vorndran J, Franke J. Microservice-based architecture for engineering tools enabling a collaborative multi-user configuration of robot-based automation solutions. *Proc CIRP*. 2019;86:86-91. http://www.sciencedirect.com/science/article/pii/S2212827120300172.
164. Elenteny R. Microservice definition and architecture; 2020. https://dzone.com/articles/microservice-definition-and-architecture.
165. Tilkov S. One size does not fit all; 2016. https://gotocon.com/dl/goto-london-2016/slides/Stefan_Tilkov-OneSizeDoesNotFitAllGOTOLondon16.pdf.
166. Medvidovic N, Taylor RN. A classification and comparison framework for software architecture description languages. *IEEE Trans Softw Eng*. 2000;26(1):70-93. https://doi.org/10.1109/32.825767.
167. Cockroft A, *State of the Art in Microservices*. Toronto, Canada: InfoQ; 2015. https://www.infoq.com/presentations/microservices-comparison-evolution.
168. Little M, *The Difference Between SOA and Microservices?*. Toronto, Canada: InfoQ; 2017. https://www.infoq.com/news/2017/07/soaandmicroservices.
169. Rademacher F, Sachweh S, Zündorf A. Differences between model-driven development of service-oriented and microservice architecture. Paper presented at: Proceedings of the 2017 IEEE International Conference on Software Architecture Workshops (ICSAW); 2017:38-45. doi:https://doi.org/10.1109/ICSAW.2017.32.

170. Yu Y, Silveira H, Sundaram M. A microservice based reference architecture model in the context of enterprise architecture. Paper presented at: Proceedings of the 2016 IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC); 2016:1856-1860. doi:https://doi.org/10.1109/IMCEC.2016.7867539.
171. Ford N, Building microservice architectures; 2018. http://nealford.com/downloads/Building_Microservice_Architectures_Neal_Ford.pdf.
172. Koutsouras A, Kougioumoutzakis D, Kantzavelou I. Assessment and security issues in cloud computing services. Paper presented at: Proceedings of the 19th Panhellenic Conference on Informatics; 2015:165-166; New York, NY, ACM. https://doi.org/10.1145/2801948.2802030
173. Dragoni N, Lanese I, Larsen ST, Mazzara M, Mustafin R, Safina L. Microservices: How To Make Your Application Scale. *CoRR*. 2017;abs/1702.07149 http://arxiv.org/abs/1702.07149.
174. Salah T, Zemerly MJ, Yeun CY, Al-Qutayri M, Al-Hammadi Y. The evolution of distributed systems towards microservices architecture. Paper presented at: Proceedings of the 2016 11th International Conference for Internet Technology and Secured Transactions (ICITST); 2016:318-325. doi:https://doi.org/10.1109/ICITST.2016.7856721.
175. N Herzberg O. Kopp J. Lenhard C Hochreiner. Analyzing the relevance of SOA patterns for microservice-based systems. Paper presented at: Proceedings of the10th ZEUS Workshop, ZEUS 2018; 2018.
176. Bakshi K. Microservices-based software architecture and approaches. Paper presented at: Proceedings of the 2017 IEEE Aerospace Conference; 2017:1-8. doi:https://doi.org/10.1109/AERO.2017.7943959.
177. Herzberg N, Hochreiner C, Kopp O, Lenhard J. Challenges of microservices architecture: a survey on the state of the practice. Paper presented at: Proceedings of the 10th ZEUS Workshop, ZEUS 2018; 2018.
178. Joselyne MI, Kanagwa B, Balikuddembe J. A framework to Modernize SME Application in Emerging Economies: Microservice Architecture Pattern Approach. *Microservices*. 2017.
179. Almeida WHC, de Aguiar Monteiro L, Hazin RR, de Lima AC, Ferraz FS. Survey on microservice architecture-security, privacy and standardization on cloud computing environment. Paper presented at: Proceedings of the 12th International Conference on Software Engineering Advances (ICSEA 2017); 2017:210.
180. Mont O, Introducing and developing a Product-Service System (PSS) concept in Sweden. ARRAY(0x8348a88); 2001.
181. Baines T. Servitization: from understanding to implementation; 2016. https://www.advancedservicesgroup.co.uk/single-post/2016/10/12/Servitization-From-understanding-to-implementation.
182. Vandermerwe S, Rada J. Servitization of business: adding value by adding services. *Eur Manag J*. 1988;6(4):314-324. http://www.sciencedirect.com/science/article/pii/0263237388900333. https://doi.org/10.1016/0263-2373(88)90033-3.
183. Baines TS, Lightfoot HW, Benedettini O, Kay JM. The servitization of manufacturing: a review of literature and reflection on future challenges. *J Manuf Technol Manag*. 2009;20(5):547-567. https://doi.org/10.1108/17410380910960984.
184. Baines T. Digitalisation and servitization: the competitive advantage? 2018. https://www.advancedservicesgroup.co.uk/single-post/2018/04/04/Digitalisation-and-servitization-the-competitive-advantage.
185. ThoughtWorks, Otto | From legacy systems to fast and flexible platforms; 2016. https://youtu.be/bSvjZi3WKZQ.
186. Godwin S. Cloud-based microservices powering BBC iPlayer; 2016. https://www.infoq.com/presentations/bbc-microservices-aws?utm_campaign=infoq_content&utm_source=infoq&utm_medium=feed&utm_term=Microservices.
187. Newman S. Practical considerations for microservice architectures. Part 1; 2014. https://www.youtube.com/watch?v=12n9E5L6qgs.
188. Bass L, Weber I, Zhu L. *DevOps: A Software Architect's Perspective*. London, United Kingdom: Pearson Education; 2015.
189. Stenberg J, *Exploring the Hexagonal Architecture*. Toronto, Canada: InfoQ; 2014. https://www.infoq.com/news/2014/10/exploring-hexagonal-architecture.
190. Klock S, Van der Werf JMEM, Guelen JP, Jansen S. Workload-based clustering of coherent feature sets in microservice architectures. Paper presented at: Proceedings of the 2017 IEEE International Conference on Software Architecture (ICSA); 2017:11-20. doi:https://doi.org/10.1109/ICSA.2017.38.
191. Cheng BC, de Lemos R, Giese H, et al. Software engineering for self-adaptive systems: a research roadmap. In: Cheng BHC, de Lemos R, Giese H, Inverardi P, Magee J, eds. *Software Engineering for Self-Adaptive Systems*. Vol 5525. Berlin, Heidelberg/Germany: Springer; 2009:1-26. https://doi.org/10.1007/978-3-642-02161-9_1.
192. Haselböck S, Weinreich R, Buchgeher G. Decision models for microservices: design areas, stakeholders, use cases, and requirements. In: Lopes A, de Lemos R, eds. *Software Architecture*. Cham: Springer International Publishing; 2017:155-170.
193. Perry DE, Porter AA, Votta LG. Empirical studies of software engineering: a roadmap. Paper presented at: Proceedings of theProceedings of the Conference on The Future of Software Engineering; 2000:345-355; New York, NY, ACM. https://doi.org/10.1145/336512.336586
194. Kitchenham BA, Dyba T, Jorgensen M. Evidence-based software engineering. Paper presented at: Proceedings of the 26th International Conference on Software Engineering; 2004:273-281; Washington, DC, IEEE Computer Society. http://dl.acm.org/citation.cfm?id=998675.999432.
195. Ponce F, Márquez G, Astudillo H. Migrating from monolithic architecture to microservices: a rapid review. Paper presented at: Proceedings of the 2019 38th International Conference of the Chilean Computer Science Society (SCCC); 2019:1-7.
196. Werner, C. M. L. A survey on microservices criticality attributes on established architectures. Paper presented at: Proceedings of the 2019 International Conference on Information Systems and Software Technologies (ICI2ST); 2019:149-155.
197. Chávez K, Cedillo P, Espinoza M, Saquicela V. A systematic literature review on composition of microservices through the use of semantic annotations: solutions and techniques. Paper presented at: Proceedings of the 2019 International Conference on Information Systems and Computer Science (INCISCOS); 2019:311-318.

198. Garriga M. Towards a taxonomy of microservices architectures. In: Cerone A, Roveri M, eds. *Software Engineering and Formal Methods*. Cham: Springer International Publishing; 2018:203-218.
199. Joseph CT, Chandrasekaran K. Straddling the crevasse: A review of microservice software architecture foundations and recent advancements. *Software: Practice and Experience*. 2019;49(10):1448-1484.
200. Cerny T, Donahoo MJ, Pechanec J. Disambiguation and comparison of SOA, microservices and self-contained systems. Paper presented at: Proceedings of the International Conference on Research in Adaptive and Convergent Systems; 2017:228-235; New York, NY, ACM. https://doi.org/10.1145/3129676.3129682.
201. Vural H, Koyuncu M, Guney S. A systematic literature review on microservices. In: Gervasi O, Murgante B, Misra S, et al., eds. *Computational Science and Its Applications – ICCSA 2017*. Cham: Springer International Publishing; 2017:203-217.
202. Soldani J, Tamburri DA, Heuvel WJVD. The pains and gains of microservices: A Systematic grey literature review. *Journal of Systems and Software*. 2018;146:215-232. http://www.sciencedirect.com/science/article/pii/S0164121218302139. https://doi.org/10.1016/j.jss.2018.09.082.
203. Dragoni N, Giallorenzo S, Lluch-Lafuente A, et al. Microservices: yesterday, today, and tomorrow. *CoRR*. 2016;abs/1606.04036 http://arxiv.org/abs/1606.04036.
204. Balalaie A, Heydarnoori A, Jamshidi P, Tamburri DA, Lynn T. Microservices migration patterns. *Softw Pract Exp*. 2018;48(11):2019-2042. https://doi.org/10.1002/spe.2608.
205. Alshuqayran N, Ali N, Evans R. Towards micro service architecture recovery: an empirical study. paper presented at: proceedings of the IEEE International Conference on Software Architecture; 2018; 2018.
206. Cerny T. Aspect-oriented challenges in system integration with microservices, SOA and IoT. *Enterpr Inf Syst*. 2018;13:467-489. https://doi.org/10.1080/17517575.2018.1462406.
207. Li S. Understanding quality attributes in microservice architecture. Paper presented at: Proceedings of the 2017 24th Asia-Pacific Software Engineering Conference Workshops (APSECW); 2017:9-10. doi:https://doi.org/10.1109/APSECW.2017.33.
208. Carlson L. What are microservices? Lightweight software development explained; 2017. https://www.infoworld.com/article/3237697/application-development/what-are-microservices-lightweight-software-development-explained.html.
209. Stenberg J, *About the SOA Heritage Impact on Microservices*. Toronto, Canada: InfoQ; 2017. https://www.infoq.com/news/2017/11/soa-impact-microservices.
210. Stenberg J, *Microservices and Teams at Amazon*. Toronto, Canada: InfoQ; 2015. https://www.infoq.com/news/2015/12/microservices-amazon.
211. Erl T. *SOA Principles of Service Design*. London, United Kingdom: Pearson Education; 2007 https://books.google.co.uk/books?id=mkQJvjR2sX0C.
212. Erl T. *Service-Oriented Architecture: Concepts, Technology, and Design*. London, United Kingdom: Pearson Education; 2005. https://books.google.co.uk/books?id=y2MALc9HOF8C.
213. MacKenzie CM, Laskey K, McCabe F, Brown PF, Metz R, Hamilton BA. Reference model for service oriented architecture 1.0. *OASIS Stand*. 2006;12:18.
214. Rosen M, Lublinsky B, Smith KT, Balcer MJ. *Applied SOA: Service-Oriented Architecture and Design Strategies*. New Jersey, United States of America: Wiley; 2012 https://books.google.co.uk/books?id=GFL9lWKojFYC.
215. Perrey R, Lycett M. Service-oriented architecture. Paper presented at: Proceedings of the 2003 Symposium on Applications and the Internet Workshops; vol 2003; 2003:116-119. doi:https://doi.org/10.1109/SAINTW.2003.1210138.
216. Zimmermann O, Doubrovski V, Grundler J, Hogg K. Service-oriented architecture and business process choreography in an order management scenario: rationale, concepts, lessons learned. Paper presented at: Proceedings of the Companion to the 20th Annual ACM SIGPLAN Conference on Object-oriented Programming, Systems, Languages, and Applications; 2005:301-312; New York, NY ACM. https://doi.org/10.1145/1094855.1094965.
217. Papazoglou MP. Service-oriented computing: concepts, characteristics and directions. Paper presented at: Proceedings of the 4th International Conference on Web Information Systems Engineering WISE 2003; 2003:3-12. doi:https://doi.org/10.1109/WISE.2003.1254461.
218. Papazoglou M, Traverso P, Dustdar S, Leymann F, Krämer BJ. Service-oriented computing research roadmap; 2006.
219. Papazoglou MP, Georgakopoulos D. Introduction: service-oriented computing. *Commun ACM*. 2003;46(10):24-28. https://doi.org/10.1145/944217.944233.
220. Channabasavaiah K, Holley K, Tuggle E. Migrating to a service-oriented architecture. *IBM DevelopWorks*. 2003;16:727-728.
221. Bianco P, Kotermanski R, Merson PF. *Evaluating a Service-Oriented Architecture*. Pennsylvania, United States of America: Carnegie Mellon University; 2007.
222. Krafzig D, Banke K, Slama D. Enterprise SOA: service-oriented architecture best practices. prentice hall professional technical reference; 2005. https://books.google.co.uk/books?id=R7oGhITYUuUC.
223. Richards M. *Microservices vs. Service-Oriented Architecture*. California, United States of America: O Reilly Media; 2016.
224. Babar MA, Dingsøyr T, Lago P, vander Vilet H. *Software Architecture Knowledge Management - Theory and Practice*. 1st ed. Berlin, Heidelberg/Germany: Springer-Verlag; 2009.
225. Newman S. *Building Microservices*. 1st ed. California, United States of America: O'Reilly Media; 2015.
226. Zachman JA. *The Zachman Framework For Enterprise Architecture: Primer for Enterprise Engineering and Manufacturing*. Colorado, United States of America: Zachman International; 2003.

227. White J, Strowd HD, Schmidt DC. Creating self-healing service compositions with feature modeling and microrebooting. *Int J Bus Process Integr Manag (IJBPIM) Special Iss Model-Driven Serv-Orient Architec*. 2009;4:35–46.
228. Vogel T, Giese H. Model-driven engineering of self-adaptive software with EUREMA. *ACM Trans Auton Adapt Syst*. 2014;8(4):18:1-18:33. https://doi.org/10.1145/2555612.
229. Danilovic M, Browning TR. Managing complex product development projects with design structure matrices and domain mapping matrices. *Int J Project Manag*. 2007;25(3):300-314. http://www.sciencedirect.com/science/article/pii/S0263786306001645. https://doi.org/10.1016/jijproman.2006.11.003.
230. Mo R, Cai Y, Kazman R, Xiao L, Feng Q. Decoupling level: a new metric for architectural maintenance complexity. Paper presented at: Proceedings of the 38th International Conference on Software Engineering; 2016:499-510.
231. Sheppard M. Design metrics: an empirical analysis. *Softw Eng J*. 1990;5(1):3-10.
232. Rombach HD. A controlled expeniment on the impact of software structure on maintainability. *IEEE Trans Softw Eng*. 1987;SE-13(3):344-354. https://doi.org/10.1109/TSE.1987.233165.
233. Briand LC, Morasca S, Basili VR. Measuring and assessing maintainability at the end of high level design. Paper presented at: Proceedings of the Conference on Software Maintenance; 1993:88-97; Washington, DC, IEEE Computer Society. http://dl.acm.org/citation.cfm?id=645542.658018.
234. Ozkaya I, Kazman R, Klein M. *Quality-Attribute-Based Economic Valuation of Architectural Patterns*. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University; 2007.
235. Sullivan KJ, Chalasani P, Jha S, Sazawal V. Software design as an investment activity: a real options perspective. *Real Options and Business Strategy: Applications to Decision Making*. Vol 10; 1999:215-262.
236. Sullivan KJ, Griswold WG, Cai Y, Hallen B. The structure and value of modularity in software design. Paper presented at: Proceedings of the 8th European Software Engineering Conference Held Jointly with 9th ACM SIGSOFT International Symposium on Foundations of Software Engineering; 2001:99-108; New York, NY, ACM. https://doi.org/10.1145/503209.503224.
237. Asundi J, Kazman R, Klein M. *Using Economic Considerations to Choose Among Architecture Design Alternatives*. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University; 2001.
238. Riegg CS, Edwin KJ. *Cost-Effectiveness and Cost-Benefit Analysis*. New York, United States of America: JohnWileySons; 2015:636-672.
239. Scaffidi, C., Arora, A., Butler, S., & Shaw, M. A value-based approach to predicting system properties from design. Paper presented at: Proceedings of the 5th EDSER; 2005.
240. Dobrica L, Niemela E. A survey on software architecture analysis methods. *IEEE Trans Softw Eng*. 2002;28(7):638-653. https://doi.org/10.1109/TSE.2002.1019479.
241. Gamma E, Helm R, Johnson R, Vlissides J. *Design Patterns: Elements of Reusable Object-Oriented Software*. London, United Kingdom: Pearson Education; 1994 https://books.google.co.uk/books?id=6oHuKQe3TjQC.
242. Cardoso J, Sheth A, Miller J, Arnold J, Kochut K. Quality of service for workflows and web service processes. *J Web Semant*. 2004;1(3):281-308: http://www.sciencedirect.com/science/article/pii/S157082680400006X. https://doi.org/10.1016/j.websem.2004.03.001.
243. Casati F, Ilnicki S, Jin L, Krishnamoorthy V, Shan MC. Adaptive and dynamic service composition in eFlow. Paper presented at: Proceedings of the 12th International Conference on Advanced Information Systems Engineering; 2000:13-31; London, UK, Springer-Verlag.
244. Sama M, Elbaum S, Raimondi F, Rosenblum DS, Wang Z. *Context-Aware Adaptive Applications: Fault Patterns and Their Automated Identification*. Piscataway, NJ: IEEE Press; 2010.
245. Mustafa O, Gomez JM. Using fuzzy clustering techniques to improve the design of microservices web applications. Paper presented at: Proceedings of the 2016 Eureka International Virtual Meeting Eureka OPTISAD; 2016.
246. Mustafa O, Gómez JM. Sustainable approach for improving microservices based web application. Paper presented at: Proceedings of the Sustainability Dialogue: International Conference on Sustainability and Environmental Management; 2017.
247. Kessler FB, ASTRO-CAptEvo; 2014. http://das.fbk.eu/astro-captevo.
248. Benatallah B, Dumas M, Sheng QZ, Ngu AHH. Declarative composition and peer-to-peer provisioning of dynamic Web services. Paper presented at: Proceedings of the Data Engineering, 2002 18th International Conference; 2002:297-308. doi:https://doi.org/10.1109/ICDE.2002.994738.
249. Maximilien EM, Singh MP. Toward autonomic web services trust and selection. Paper presented at: Proceedings of the 2nd International Conference on Service Oriented Computing; 2004:212-221. https://doi.org/10.1145/1035167.1035198.
250. Wang P, Chao KM, Lo CC, Farmer R, Kuo PT. A reputation-based service selection scheme. Paper presented at: Proceedings of the e-Business Engineering, 2009. ICEBE '09. IEEE International Conference; 2009:501-506.
251. Limthanmaphon B, Zhang Y. Web service composition with case-based reasoning. Paper presented at: Proceedings of the 14th Australasian Database Conference - Volume 17 Darlinghurst; 2003:201-208; Australia, Australia, Australian Computer Society, Inc. http://dl.acm.org/citation.cfm?id=820085.820122.
252. Filieri A, Ghezzi C, Tamburrelli G. A formal approach to adaptive software: continuous assurance of non-functional requirements. *Formal Aspects of Computing*. 2011;24(2):163-186. https://doi.org/10.1007/s00165-011-0207-2.
253. Calinescu R, Rafiq Y, Johnson K, Bakir ME. Adaptive model learning for continual verification of non-functional properties. Paper presented at: Proceedings of theProceedings of the 5th ACM/SPEC International Conference on Performance Engineering; 2014:87-98.
254. Calinescu R, Ghezzi C, Kwiatkowska M, Mirandola R. Self-adaptive software needs quantitative verification at runtime. *Commun ACM*. 2012;55(9):69-77. https://doi.org/10.1145/2330667.2330686.

255. Bencomo N, Belaggoun A, Issarny V. Dynamic decision networks for decision-making in self-adaptive systems: a case study. Paper presented at: Proceedings of the 8th International Symposium on Software Engineering for Adaptive and Self-Managing Systems; 2013:113-122; Piscataway, NJ, IEEE Press. http://dl.acm.org/citation.cfm?id=2663546.2663565.
256. Blair G, Bencomo N, France RB. Models@ run.time. *Computer*. 2009;42(10):22-27. https://doi.org/10.1109/MC.2009.326.
257. Zhang J, Cheng BHC. Model-based development of dynamically adaptive software. Paper presented at: Proceedings of the 28th International Conference on Software Engineering; 2006:371-380; New York, NY, ACM. https://doi.org/10.1145/1134285.1134337
258. Aßmann U, Götz S, Jézéquel JM, Morin B, Trapp M. *Models@ run. time*. Cham: Springer International Publishing; 2014:1-18.
259. Oreizy P. On the role of software architectures in runtime system reconfiguration. Paper presented at: Proceedings of the IEE Proceedings – Software; vol 145 October 1998137-145. http://digital-library.theiet.org/content/journals/10.1049/ip-sen_19982296.
260. Barbier F. MDE-based design and implementation of autonomic software components. Paper presented at: Proceedings of the 2006 5th IEEE International Conference on Cognitive Informatics; vol. 1; 2006:163-169. doi:https://doi.org/10.1109/COGINF.2006.365692.
261. Joshi KR, Hiltunen MA, Sanders WH, Schlichting RD. Automatic model-driven recovery in distributed systems. Paper presented at: Proceedings of the24th IEEE Symposium on Reliable Distributed Systems (SRDS'05); 2005:25-36. doi:https://doi.org/10.1109/RELDIS.2005.11.
262. Fleurey F, Dehlen V, Bencomo N, Morin B, Jézéquel JM. *Models in Software Engineering*. Berlin, Heidelberg/Germany: Springer-Verlag; 2009:97-108.
263. Morin B, Barais O, Nain G, Jezequel JM. Taming dynamically adaptive systems using models and aspects. Paper presented at: Proceedings of the 2009 IEEE 31st International Conference on Software Engineering; 2009:122-132.
264. Floch J, Hallsteinsen S, Stav E, Eliassen F, Lund K, Gjorven E. Using architecture models for runtime adaptability. *IEEE Softw*. 2006;23(2):62-70.
265. Mongiello M, Colucci S, Vogli E, Grieco LA, Sciancalepore M. Run-time architectural modeling for future internet applications. *Complex Intell Syst*. 2016;2(2):111-124. https://doi.org/10.1007/s40747-016-0020-x.
266. Heinrich R, Zirkelbach C, Jung R. Architectural runtime modeling and visualization for quality-aware DevOps in cloud applications. Paper presented at: Proceedings of the 2017 IEEE International Conference on Software Architecture Workshops (ICSAW); 2017:199-201. doi:https://doi.org/10.1109/ICSAW.2017.33
267. van Engelen R. Code Generation Techniques for Developing Light-weight XML Web Services for Embedded Devices. Paper presented at: Proceedings of the 2004 ACM Symposium on Applied Computing; 2004:854-861; New York, NY, ACM. doi:https://doi.org/10.1145/967900.968075.
268. Kwon YW, Tilevich E. Cloud refactoring: automated transitioning to cloud-based services. *Automated Software Engineering*. 2013;21(3):345-372. https://doi.org/10.1007/s10515-013-0136-9.
269. Bencomo N, Belaggoun A. Supporting decision-making for self-adaptive systems: from goal models to dynamic decision networks. In: Doerr J, Opdahl A, eds. *Requirements Engineering: Foundation for Software Quality*. Vol 7830. Berlin Heidelberg/Germany: Springer; 2013:221-236. https://doi.org/10.1007/978-3-642-37422-7_16.
270. Cheng SW, Garlan D, Schmerl B. Self-star properties in complex information systems. In: Babaoglu O, Jelasity M, Montresor A, Fetzer C, Leonardi S, eds. *Making Self-adaptation an Engineering Reality*. Berlin, Heidelberg: Springer-Verlag; 2005:158-173 http://dl.acm.org/citation.cfm?id=2167575.2167589.
271. Garlan D, Cheng SW, Huang AC, Schmerl B, Steenkiste P. Rainbow: architecture-based self-adaptation with reusable infrastructure. *Computer*. 2004;37(10):46-54. https://doi.org/10.1109/MC.2004.175.
272. IBM. *An Architectural Blueprint for Autonomic Computing*. Indiana, United States of America: IBM; 2006.
273. de Lemos R, Giese H, Müller H, et al. Software engineering for self-adaptive systems: a second research roadmap. In: de Lemos R, Giese H, Muller H, Shaw M, eds. *Software Engineering for Self-Adaptive Systems II*. Vol 7475. Berlin Heidelberg: Springer; 2013:1-32. https://doi.org/10.1007/978-3-642-35813-5_1.
274. Dobson S, Denazis S, Fernández A, et al. A Survey of Autonomic Communications. *ACM Trans Auton Adapt Syst*. 2006;1(2):223-259. https://doi.org/10.1145/1186778.1186782.
275. Burns R. *Advanced Control Engineering*. Oxford, United Kingdom: Butterworth-Heinemann; 2001 https://books.google.co.uk/books?id=DovwVQu6ImsC.
276. Cámara J, Garlan D, Kang WG, Peng W, Schmerl B. *Uncertainty in Self-Adaptive Systems Categories, Management, and Perspectives*. Pennsylvania, United States of America: Carnegie Mellon University; 2017.
277. Georgiadis I, Magee J, Kramer J. Self-organising software architectures for distributed systems. Paper presented at: Proceedings of the 1st Workshop on Self-healing Systems; 2002:33-38; New York, NY, ACM. doi:https://doi.org/10.1145/582128.582135.
278. Malek S, Mikic-Rakic M, Medvidovic N. A decentralized redeployment algorithm for improving the availability of distributed systems. In: Dearle A, Eisenbach S, eds. *Component Deployment*. Vol 3798. Berlin, Heidelberg/Germany: Springer; 2005:99-114. https://doi.org/10.1007/11590712_8.
279. Vromant P, Weyns D, Malek S, Andersson J. On interacting control loops in self-adaptive systems. Paper presented at: Proceedings of the 6th International Symposium on Software Engineering for Adaptive and Self-Managing Systems; 2011:202-207. https://doi.org/10.1145/19880081988037
280. Weyns D, Malek S, Andersson J. On decentralized self-adaptation: lessons from the trenches and challenges for the future. Paper presented at: Proceedings of the 2010 ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems; 2010:84-93; New York, NY, ACM. doi:https://doi.org/10.1145/18089841808994.

281. Dietrich J, McCartin C, Tempero E, Shah SA. Barriers to modularity - an empirical study to assess the potential for modularisation of java programs. In: Heineman G, Kofron J, Plasil F, eds. *Research into Practice - Reality and Gaps*. Vol 6093. Berlin, Heidelberg/Germany: Springer; 2010:135-150. https://doi.org/10.1007/978-3-642-13821-8_11.
282. Zimmermann O. Architectural refactoring for the cloud: a decision-centric view on cloud migration. *Computing*. 2017;99(2):129-145. https://doi.org/10.1007/s00607-016-0520-y.
283. Losavio F, Ordaz O, Esteller V. Refactoring-based design of reference architecture. *Revista Antioqueña de las Ciencias Computacionales*. 2015;5(1).32–48.
284. Stal M. Chapter 3 - refactoring software architectures. In: Babar MA, Brown AW, Mistrik I, eds. *Agile Software Architecture*. Boston, MA: Morgan Kaufmann; 2014:63-82. https://www.sciencedirect.com/science/article/pii/B9780124077720000034. https://doi.org/10.1016/B978-0-12-407772-000003-4.
285. Schmidt F, MacDonell SG, Connor AM. In: Lee R, ed. *An Automatic Architecture Reconstruction and Refactoring Framework*. Berlin, Heidelberg/Germany: Springer; 2012:95-111. https://doi.org/10.1007/978-3-642-23202-2_7.
286. Jamshidi P, Pahl C, Chinenyeze S, Liu X. Cloud Migration Patterns: A Multi-cloud Service Architecture Perspective. In: Toumani F, Pernici B, Grigori D, et al., eds. *Service-Oriented Computing - ICSOC 2014 Workshops*. Cham: Springer International Publishing; 2015:6-19.
287. Zimmermann O. Architectural decision identification in architectural patterns. Paper presented at: Proceedings of the WICSA/ECSA 2012 Companion; 2012:96-103; New York, NY, ACM. doi:https://doi.org/10.1145/23619992362021.
288. Oreizy P, Medvidovic N, Taylor RN. Architecture-based runtime software evolution. Paper presented at: Proceedings of the 20th International Conference on Software Engineering; 1998:177-186; Washington, DC, IEEE Computer Society. http://dl.acm.org/citation.cfm?id=302163.302181.
289. Ivkovic I, Kontogiannis K. A framework for software architecture refactoring using model transformations and semantic annotations. Paper presented at: Proceedings of the Conference on Software Maintenance and Reengineering (CSMR'06); 2006:10. doi:https://doi.org/10.1109/CSMR.2006.3.
290. McIlraith S, Son T. Adapting golog for composition of semantic web services. Paper presented at: Proceedings of the Eighth International Conference on Knowledge Representation and Reasoning (KR2002); 2002:482-493; Toulouse, France.

How to cite this article: Hassan S, Bahsoon R, Kazman R. Microservice transition and its granularity problem: A systematic mapping study. *Softw Pract Exper*. 2020;50:1651–1681. https://doi.org/10.1002/spe.2869